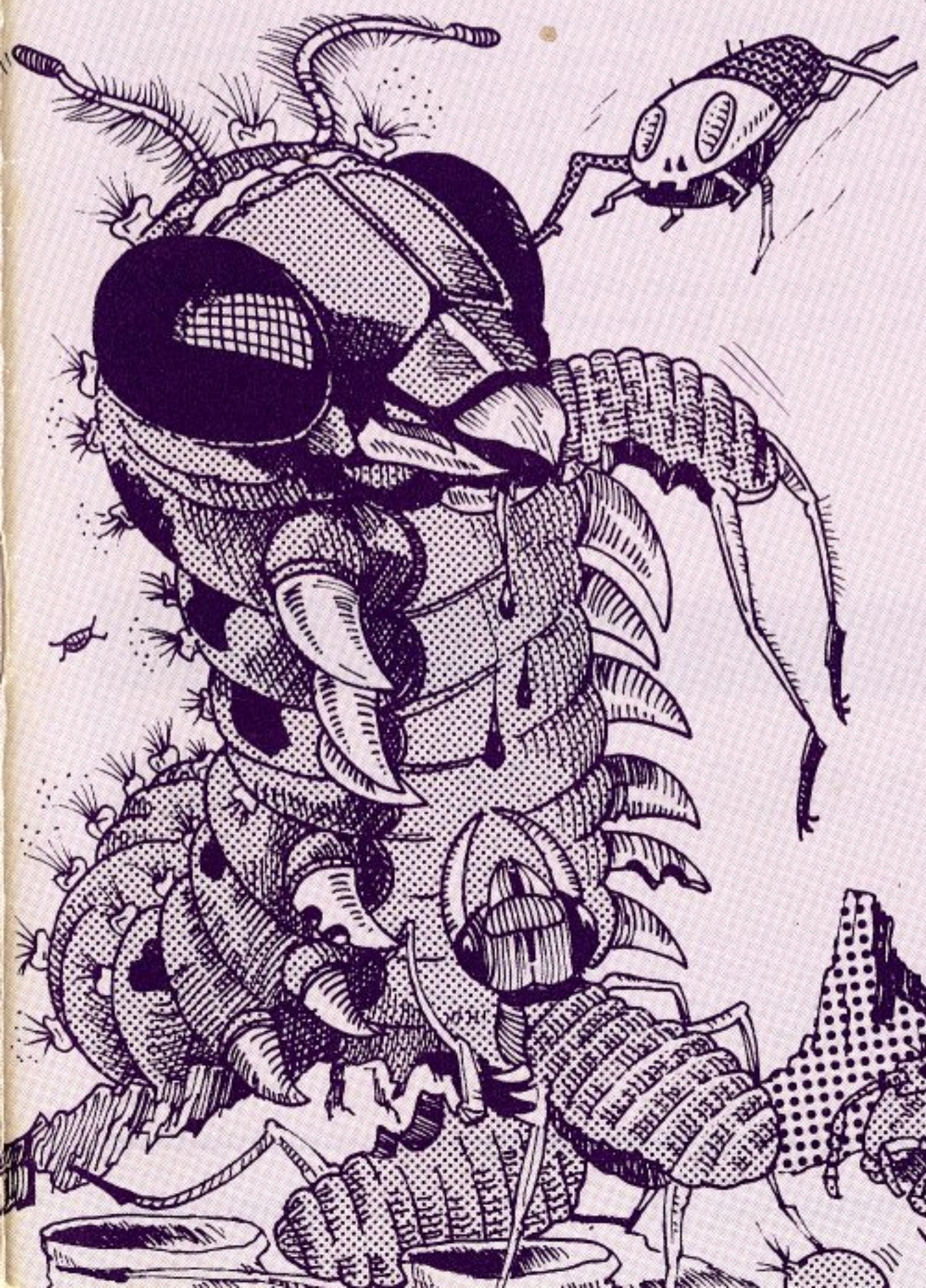
£1.00

ELBUG FOR THE ELECTRON

Caterpillar

Vol 2 No 3 JAN/FEB 1985



GAMES

* Caterpillar
* Patience

PLUS

* Making Tunes on the Electron
* Mini Text Editor Extended
* 3D Surfaces
* File Handling

PLUS

* Adventure Games
* Graphics Packages Reviewed
* Games Reviews
* And much more

EDITORIAL

THIS MONTH'S MAGAZINE

A major feature in this month's issue is a much extended version of the Mini Text Editor first published in ELBUG Vol.1 No.9 (Aug/Sept 1984). This enhanced version provides both right and left justification of text just like a real word processor, and also allows you to set up tab positions, most useful for tables and special layouts.

A seldom covered aspect of the Electron is its sound capability. Following the article last month on sychronizing words and music, we present the first of a two part article that shows you how to use your Electron for playing tunes.

In the past we have published articles on the subject of debugging programs, something dear to the heart of all Basic programmers. One aid to this is a 'Trace' facility. Our utility this month provides a vast improvement on the built in trace in BBC Basic and turns this little used feature into something really useful.

We also feature this month two more excellent games for the Electron. Caterpillar, featured on the front cover of this issue, has some of the fastest action we've seen in an ELBUG game, while Patience is a very appealing implemetation of this popular card game, with the computer doing all the work of shuffling, dealing the cards and turning them over.

COMPETITION

Graphics is something that interests many micro owners, and we are sure the 3D surfaces program published in this issue will appeal to many with its impressive displays. Not only that, but you can readily incorporate and display a surface of your own choosing. In conjunction with our magazine for BBC micro owners, BEEBUG, we are offering a prize of £50 for the best and most interesting display sent in. Full details are contained with the article itself.

NEXT ISSUE

Remember that this is a two month issue (January/February). The next issue will be the March edition which should be with you by the end of February.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

HINT WINNERS

This time we have awarded one prize of £15 to D.Morgan who has contributed several of the hints published in this issue of ELBUG. Further hints and tips for the Electron are always welcome.

MAGAZINE CASSETTE

Once again all the programs featured in this issue of ELBUG are available on cassette ready to load straight into your Electron. We have also included the small data file created by last month's file handling programs ready to use with the programs included with this month's article on the same subject.

ELBUG MAGAZINE

GENERAL CONTENTS

PROGRAMS

HINTS, TIPS & INFO

MAKING TUNES ON THE ELECTRON (Part 1)

Charles Francis

Making music on your computer is not as difficult as you might think. Charles Francis takes baton in hand and conducts us through the world of Electron music.

Despite its limitations the music synthesiser of the Electron can play almost any tune. This article shows how, even if you have no musical knowledge, you can program the Electron to play a simple tune. You can use a musical score or write your own ditty. Tunes can be used either on their own, or to embellish a game. In a second article I will show how quite sophisticated melodies can be played.

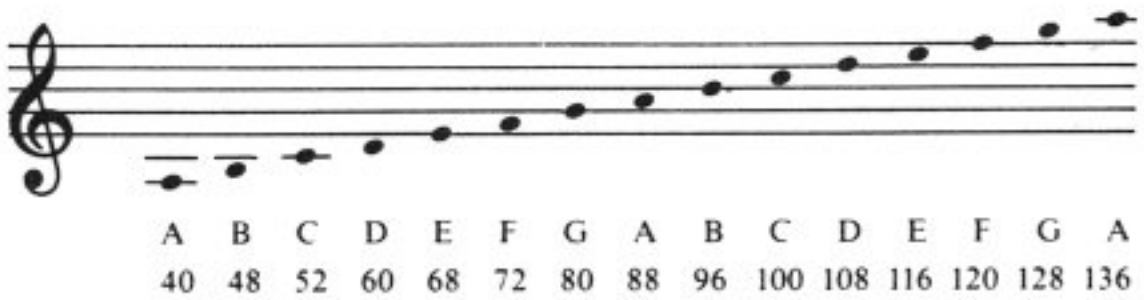


Figure 1.

Let's start with the simple scale, shown above. The letters give the notes, and the number is the P value, used in the SOUND command. The complete range of P values is given on page 118 of the User Guide. The scale shown here is the key of C natural (i.e. doh-re-mi-fah-soh-la-ti-doh which starts and ends with C as doh). I will explain how to convert to other keys later. The user guide states that P values greater than 100 are not accurate to the scale, but in practice most users will be satisfied with the range of notes given here for writing tunes.

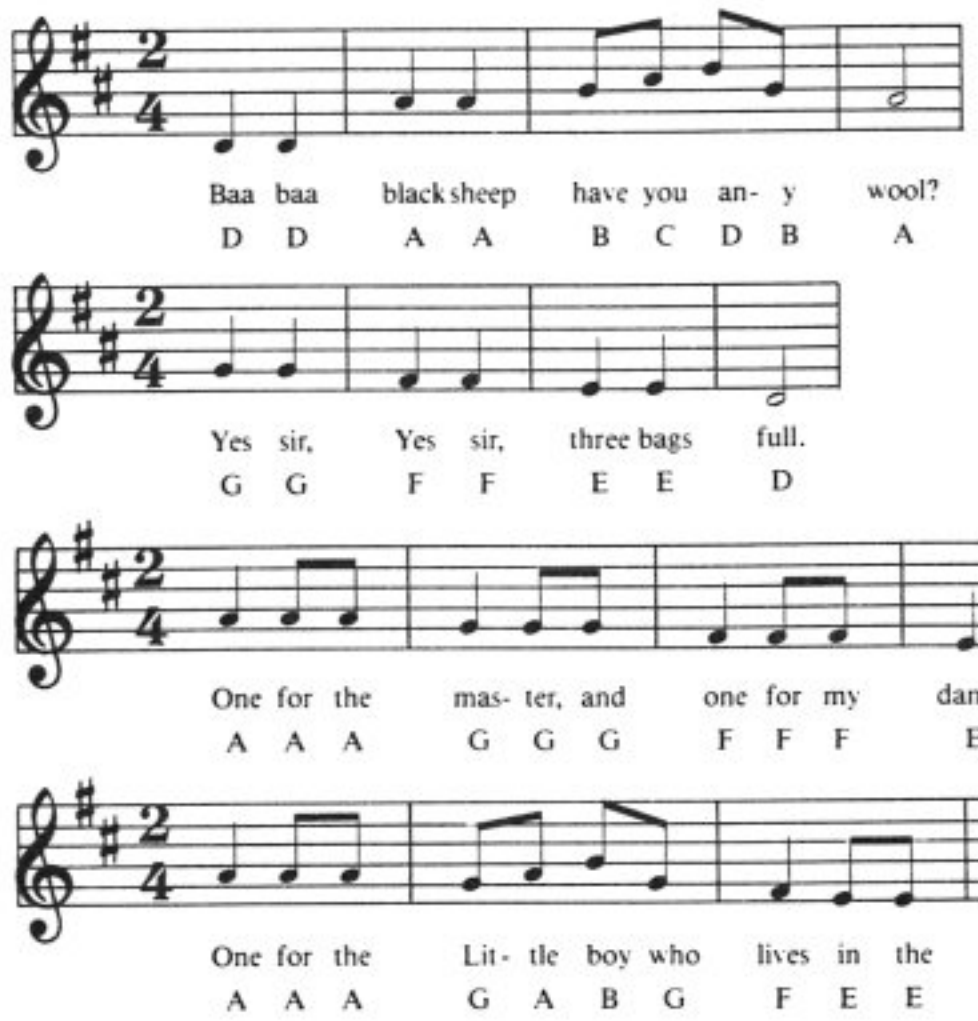


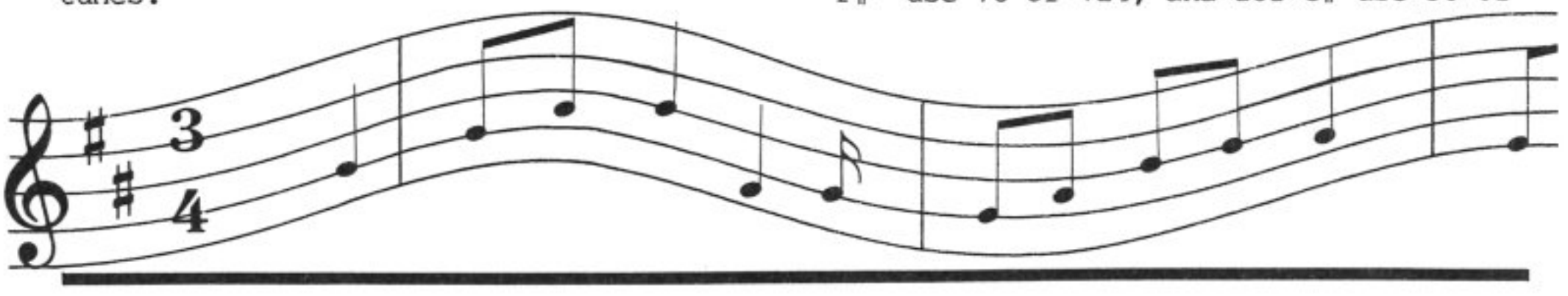
Figure 2.

Now take a simple tune. The music for 'Baa Baa, Black Sheep' is given in the diagram. This is a very useful tune for our purpose as it is well known and illustrates almost everything we need.

PITCH

The position of a note symbol on the set of lines (the stave) indicates the pitch of the note. You can work out the pitch value required to produce the note in a Basic program by comparing the position with the whole scale shown above and reading off the P value.

At the beginning of the line there are two sharp signs, #, in the positions of F and C. this means that every time an F or a C is indicated, F# or C# should be played. This is done by adding 4 to the relevant P value - for F# use 76 or 124, and for C# use 56 or



104. These changes gives the key of D, in which D is doh. In practice you don't have to know which key a tune is written in - you simply add 4 to the P values of those notes indicated by a #.

If a flat sign (b) is shown at the beginning of the tune then subtract 4 from the 'P' value of every appearance of that note. If # or b is shown by a particular note in the tune then 4 is added to or subtracted from the 'P' value for that note only. The 'natural' sign (♮) indicates that the note by which it appears should not be played sharp or flat, so 4 is not added to or subtracted from that appearance of the note.

DURATION

Each note in the tune has a particular length to give the tune its character. The length of the notes are shown by the symbol on the stave. The type of symbol on the stave will help determine the duration figure in the SOUND statement.

Note	Rest	Name	Relative Length (L%)
𝅝	𝄻	whole note (semi-breve)	16
𝅗𝅥	𝄼	half note (minim)	8
♩	𝄽 or 𝄾	quarter note (crotchet)	4
♪	𝄿	eighth note (quaver)	2
𝅘𝅥𝅯	𝅀	sixteenth note (semi-quaver)	1

A rest is treated exactly like a note but no sound is actually made.

In the arrangement of 'Baa Baa Black Sheep above, there is a dot after the note for 'dame'. This note is a quarter note, of relative length 4. The dot means that it should be held for half as long again, so that its relative length with the dot is 6.

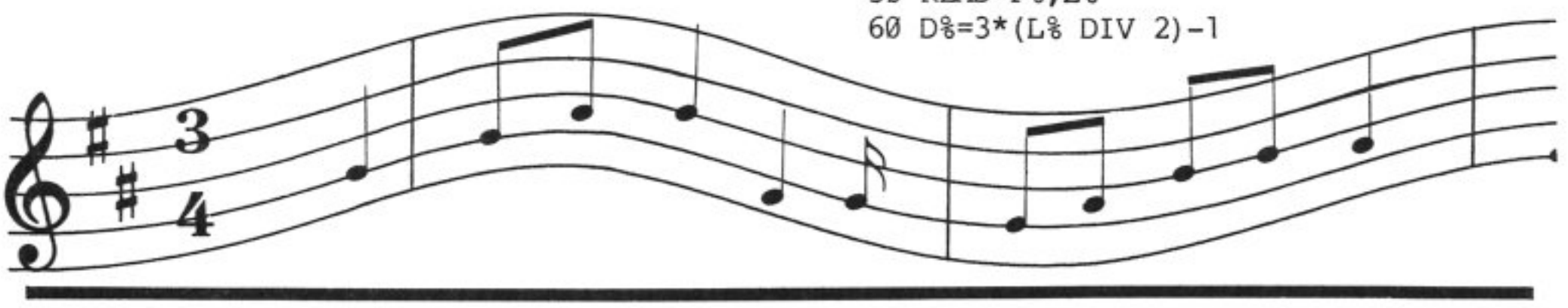
The time signature shown for 'Baa Baa Black Sheep is 2:4. This means that the tune is divided into groups of notes, called bars, each group having the same length as 2 quarter notes. In theory the first note of each group should be emphasised, but there is no easy way of doing this on the Electron, and most simple tunes survive quite well without.

TEMPO

Now it is time to choose a speed for the tune. This determines the D values used in the SOUND command. D values should be multiples of the relative lengths of notes, shown in figure 3. In 'Baa Baa, Black Sheep there are no 16th notes. A lively version results by using D=3 for eighth notes, then D=6 for quarter notes, D=12 for half notes and D=24 for whole notes.

In practice it is also necessary to put gaps between the notes. If this is not done 'Baa Baa' - both played on the same note - will sound as a single long note. So will 'Black Sheep' and the result will be terrible. SOUND 1,0,0,1 gives a convenient gap, and works well for simple tunes. The duration of the gap (D=1) must be subtracted from the D value of the note immediately preceding it, or the rhythm will be lost. We are now ready to write a program to play 'Baa Baa Black Sheep'.

```
10 REM BAA BAA BLACK SHEEP
20 P%=0:D%=0
30 REPEAT
40 SOUND 1,-15,P%,D%:SOUND 1,0,0,1
50 READ P%,L%
60 D%=3*(L% DIV 2)-1
```



```
  70 UNTIL L%<0
  80 END
  90 REM NOTE DATA: PITCH, LENGTH
 100 DATA 60,4,60,4,88,4,88,4,96,2,104
,2,108,2,96,2,88,8
 110 DATA 80,4,80,4,76,4,76,4,68,4,68,
4,60,8,88,4,88,2,88,2
 120 DATA 80,4,80,2,80,2,76,4,76,2,76,
2,68,6,68,2,88,4,88,2,88,2
 130 DATA 80,2,88,2,96,2,80,2,76,4,68,
2,68,2,60,8
 140 DATA -1,-1
```

For ease of data entry, the relative lengths of the notes are entered as L% - the note fraction (e.g. 2 for half, 4 for quarter etc). Line 60 calculates the duration value, D%, and determines the speed at which the tune is played. Experiment by changing the number 3 in line 60 to other values. Another interesting experiment is to use a gap length of zero, by putting SOUND 1,0,0,0 in line 40. If this is done, 1 should not be subtracted from the value of D% in line 60. An interesting percussive effect can be obtained by using SOUND1,-15,12,0 for the gap.

It is very simple to change key, or to tune the Electron to another instrument by inserting:

```
55 P%=P%+(number)
```

For example P%=P%-8 gives the key of C.

WRITING YOUR OWN DITTY

It is remarkably easy to write a simple ditty to embellish a game, and you do not need any musical knowledge, provided you stick to a few simple rules.

1. Keep to a recognised key - a key is a set of notes which are known to sound well together. For the Electron's synthesiser - a single note at a time- one key is much like another. You will do well to keep to the key of C, shown in figure 1 (No sharps or flats).

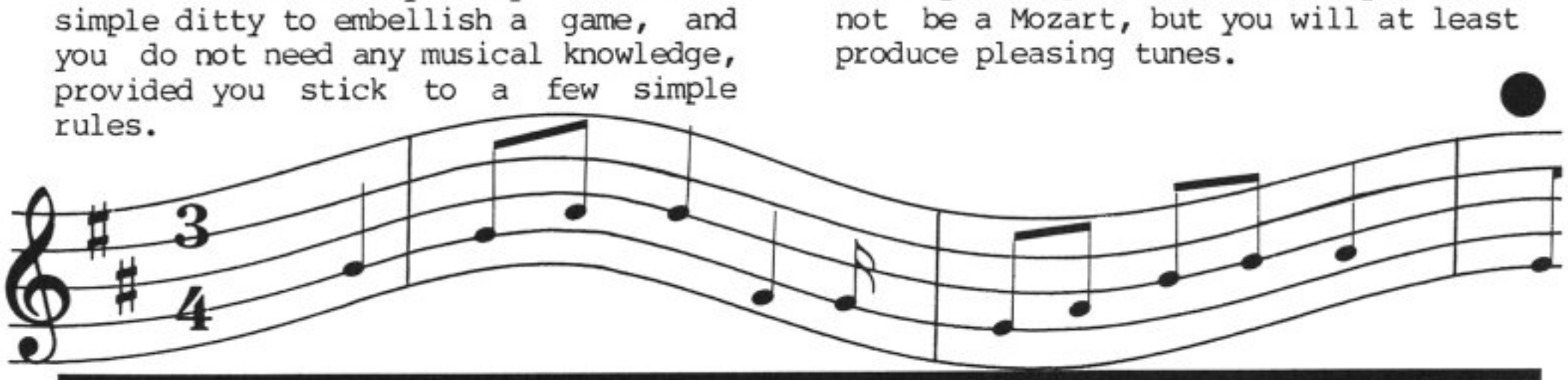
2. Keep to a time signature. Most music is written in 'common time', or 4:4. This means that notes are grouped into bars, each of length equal to four quarter notes. You could also experiment with 4:4, which is waltz time, or jig time (depending mainly on speed.) But as the Electron cannot emphasize the first note of each bar the way a musician does you would be wise not to venture outside these two signatures.

3. Use a simple rhythm, and repeat a similar rhythm pattern within each bar. It would be unwise to try a pattern in which the first note of each bar was not the longest used in the bar.

4. Be prepared to experiment. Within a particular key certain combinations sound better than others. You will soon learn which to avoid.

5. Sign off on a note that sounds well. You will have to experiment to find how to finish a particular ditty. Don't leave it hanging in mid-air, waiting for another note.

If you keep to these rules you will not be a Mozart, but you will at least produce pleasing tunes.



HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

INDIRECTED DATA - D. Morgan

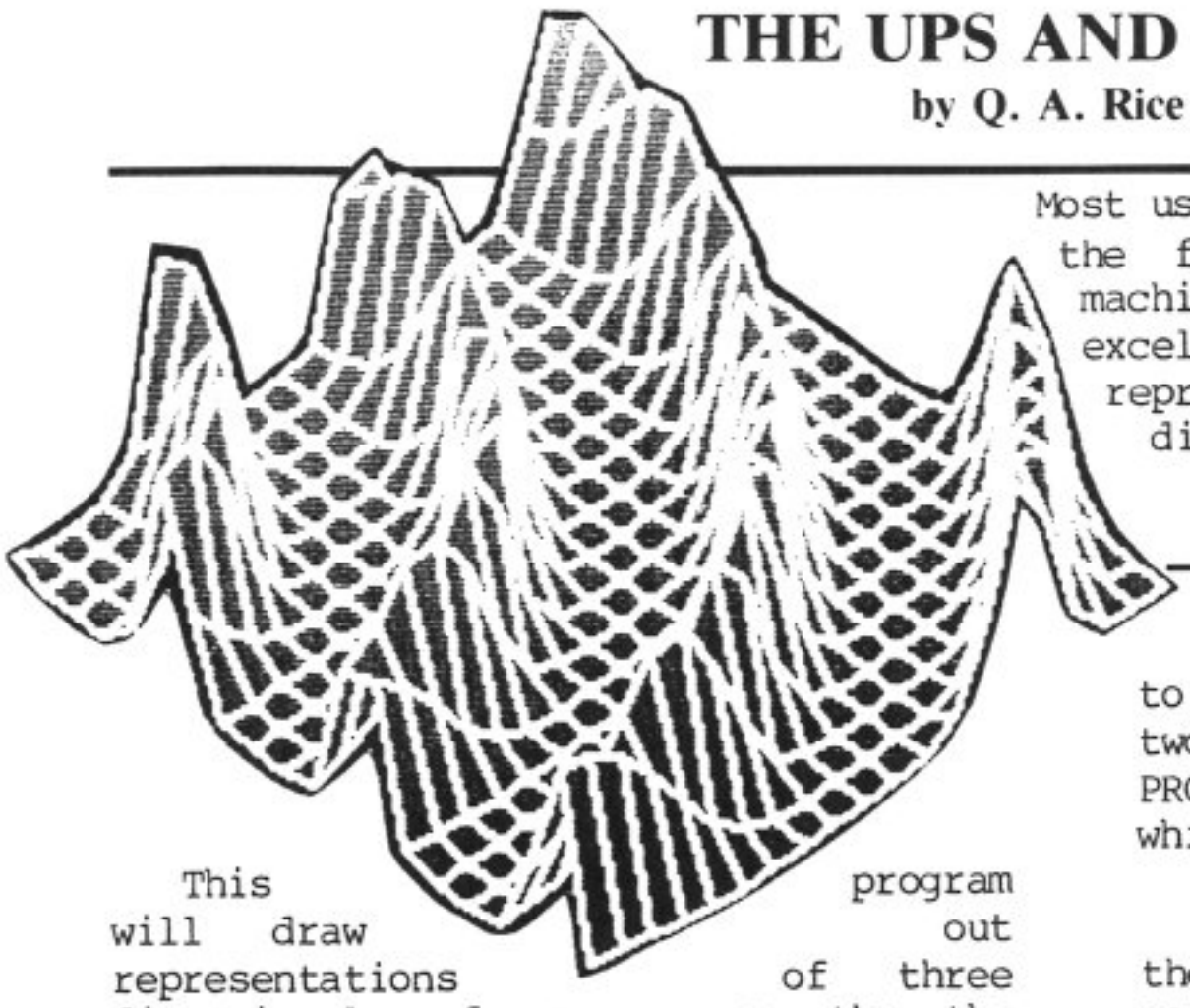
If you ever find yourself in the situation of needing to read a lot of data into memory using indirection operators, then it is worthwhile knowing that the 'memory access' can be within the READ statement. For example:

```
RESTORE:FORI%=0 TO 99
READ I%?memoryarea:NEXT
```

where 'memoryarea' is the address of an area in memory that we have previously assigned to this variable.

THE UPS AND DOWNS OF 3D SURFACES

by Q. A. Rice



Most users of the Electron will never tire of the fascinating graphics displays that this machine is capable of. This program is an excellent example of the old idea of representations of distorted three dimensional surfaces on your computer's screen.

This program will draw out representations of three dimensional surfaces on the the screen of your Electron. The surfaces follow one of a set of equations. A choice of eight equations is included in the program and there is room for your own as well. Because of the way that the surfaces are drawn (from the back forwards) a simple form of 'hidden line removal' is achieved. This makes the surfaces very realistic.

In addition the display can be drawn with perspective or not, as you choose, and in an inverted form if you so wish. The whole program is menu driven and is both simple to use and a pleasure to watch.

Type in the program carefully and then save it before you run it.

The program gives you a choice of eight different equations for the surfaces. The ninth option is for your own equation. This should be entered in line 1860 with Z as a function of X% and Y%. The equation already included in the program listing at this line just draws a flat surface. Try variations of the other equations given in lines 1780 to 1850 to get the hang of using this facility.

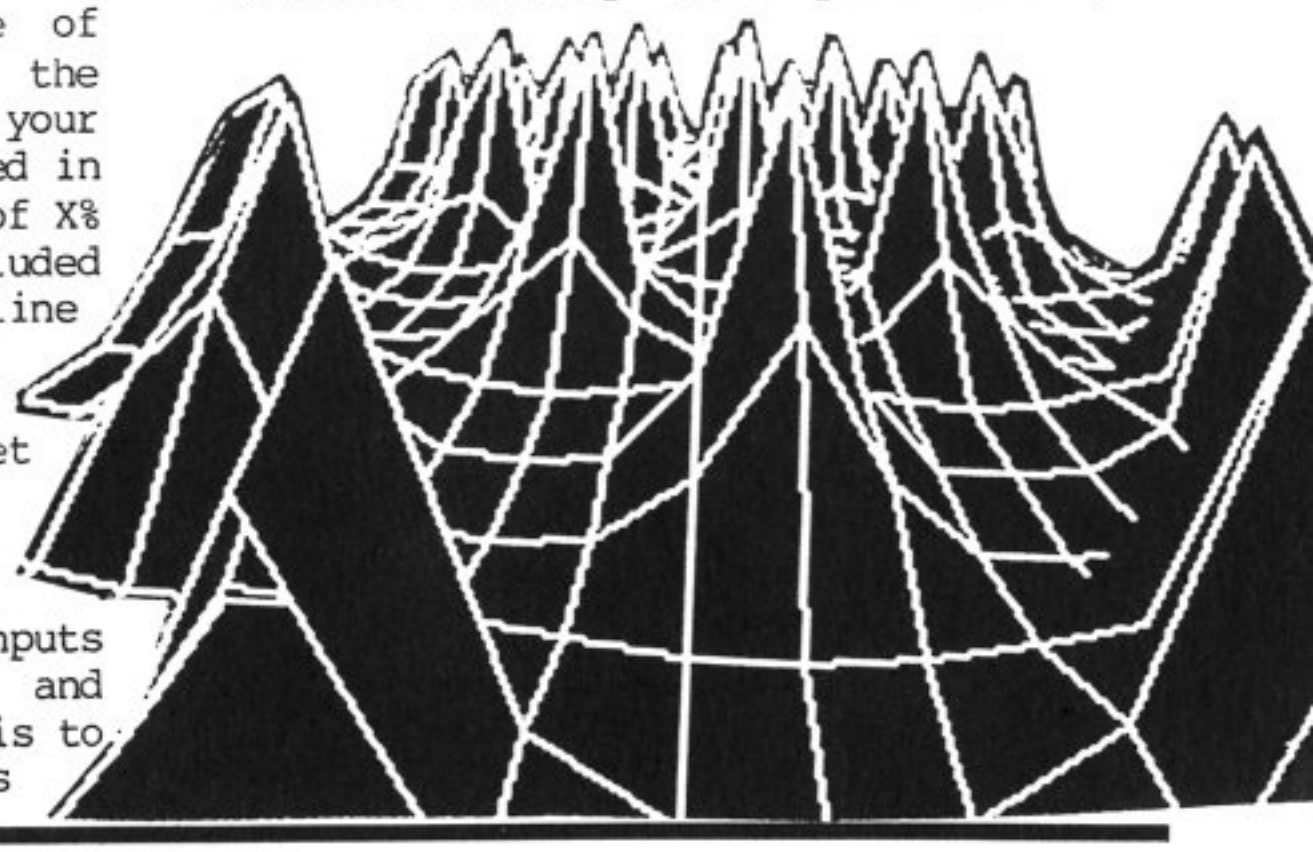
PROGRAM NOTES

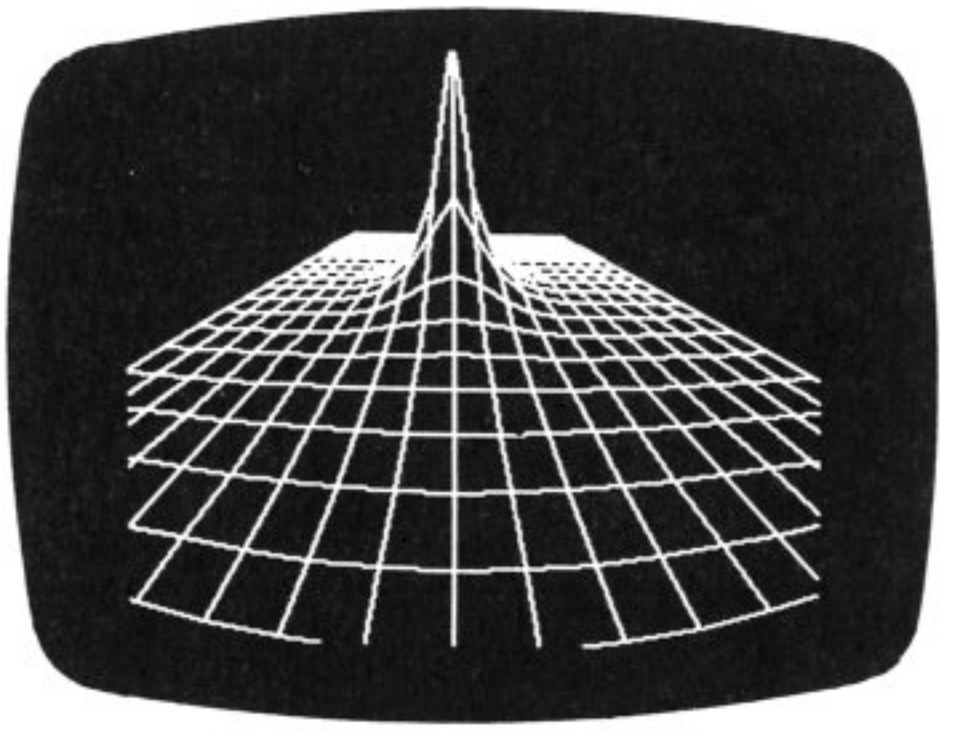
The procedure, PROCchoices, inputs from the user the equation number and the choice of whether the display is to be in perspective and whether it is to be inverted or not. Then one of the two procedures, PROCperscalc or PROCisocalc, is called, depending on which option was chosen.

The procedure chosen, calculates all the points to be plotted, using the equation chosen (PROCfnz) and then the corresponding 'draw' procedure plots the series of squares onto the screen using the common procedure, PROCdraw.

There are no hard and fast rules as to making up your own equations to enter as the ninth option. Just try whatever seems best and you will be pleasantly surprised by the elegant and vivid results.

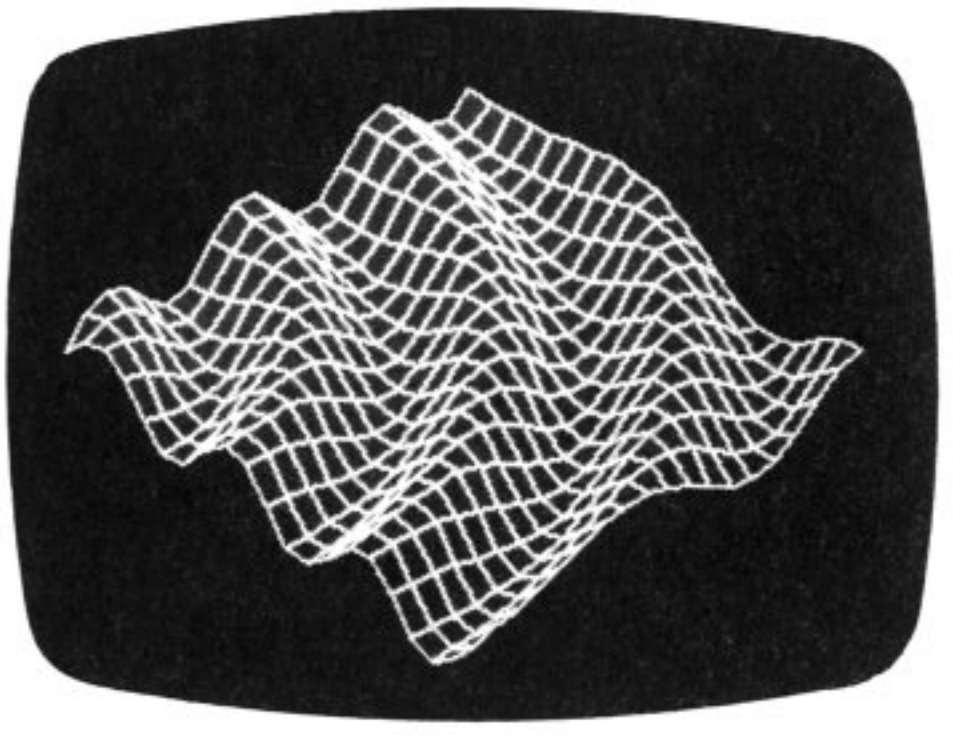
When you have experimented with this program for a while, you might like to try it with a more competitive flavour. We are offering £50 for the equation, to fit into this program at line 1860, that produces the most pleasing display. Send in your ideas (only one per person please) to the editorial address. Clearly mark your envelope

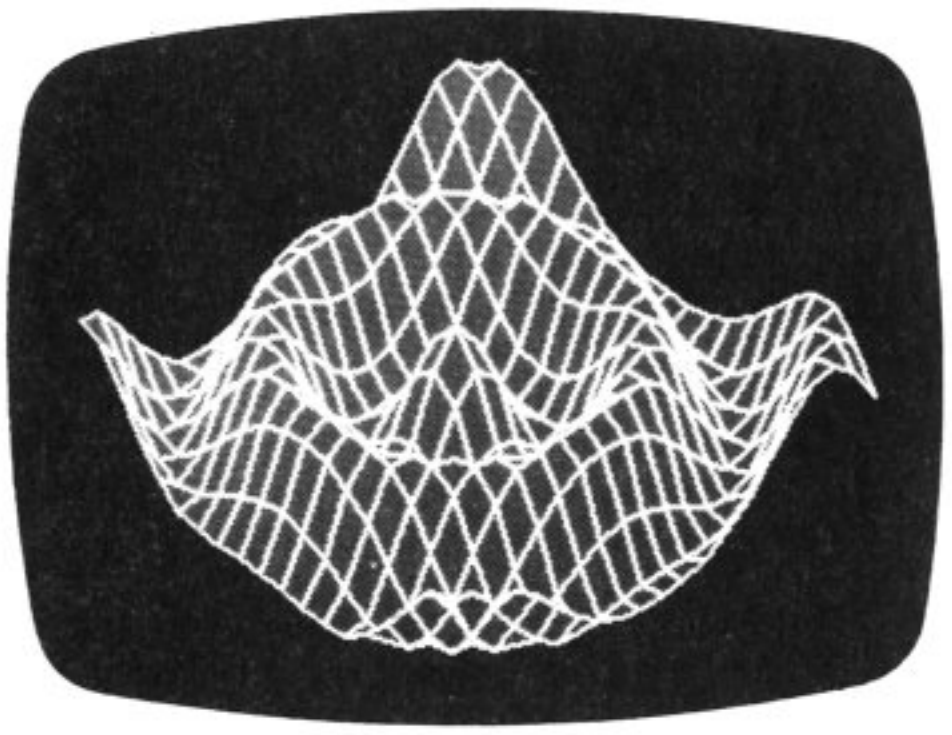




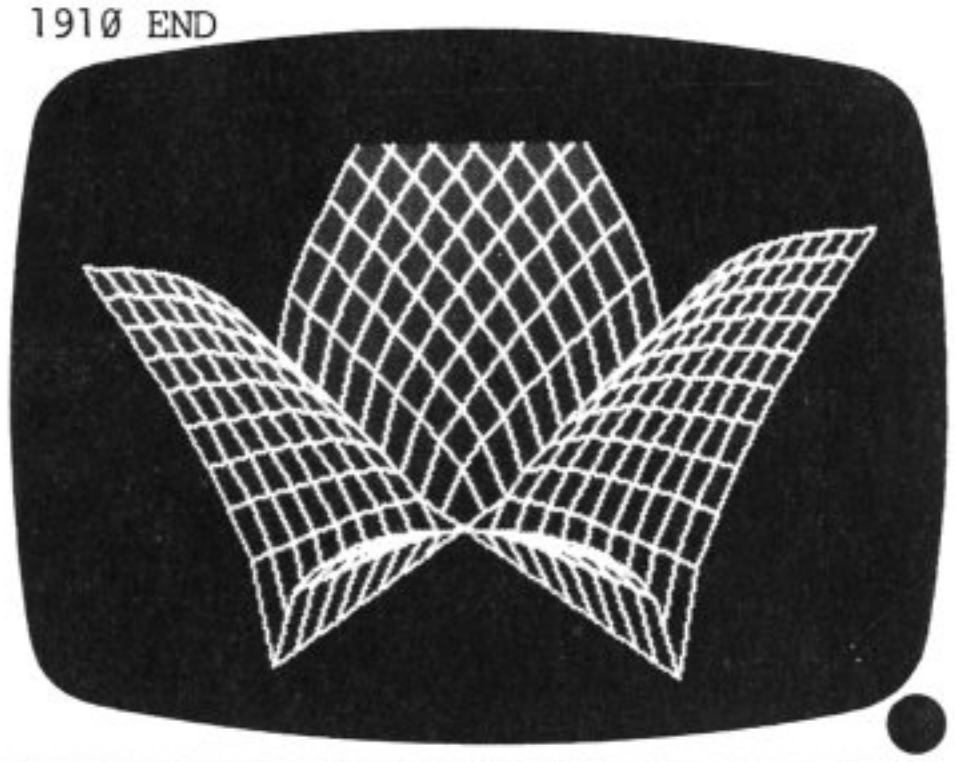
'Surface Competition'. The closing date for entries is 8th February 1985 so get your ideas in soon. The judges' decision, as they say, will be final.

```
 10 REM PROGRAM 3D SURFACE
 20 REM VERSION E2.0
 30 REM AUTHOR  Q.A.RICE
 40 REM ELBUG   JAN/FEB 1985
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
 70 ON ERROR GOTO 1870
 80 :
 90 DIM P%(20,20,2)
100 REPEAT
110 MODE 6
120 PROCchoices
130 IF PM=2 THEN PROCperscalc ELSE PR
OCisocalc
140 MODE 1:VDU 23,1,0;0;0;0;
150 IF PM=2 THEN PROCpersdraw ELSE PR
OCisodraw
160 REPEAT UNTIL GET=32
170 UNTIL FALSE
180 END
190 :
1000 DEF PROCchoices
1010 VDU23,1,0;0;0;0;
1020 PRINT TAB(9,1)"HIDDEN LINE GRAPHS
"TAB(9,2)"BLACKOUT TECHNIQUE"
1030 PRINT TAB(10,23)"ENTER CHOICE"
1040 VDU 28,8,22,39,5
1050 PRINT "1.   SQR(X)*SQR(Y)"''"2.
 COS(X)*COS(Y) #1"''
1060 PRINT "3.   COS(X)*COS(Y) #2"''"4
.   COS(X)*COS(Y)*DISTANCE"'
1070 PRINT "5.   COS(D)*COS(D)"''"6.
 EXP(DISTANCE)"'
1080 PRINT "7.   COS(CORNER DISTANCE)"
''"8.   EXP(COS(CORNER DISTANCE))"'
1090 PRINT "9.   YOUR OWN PREDEFINED"
1100 REPEAT C=(GET-48):UNTIL C>0 AND C
<10
1110 CLS
1120 PRINT "1.   ISOMETRIC"''"2.   PE
RSPECTIVE"
1130 REPEAT PM=(GET-48):UNTIL PM=1 OR
PM=2
1140 CLS
1150 PRINT "1.   NORMAL"''"2.   INVER
TED"
1160 REPEAT P=(GET-48):UNTIL P=1 OR P=2
1170 VDU 26,12
1180 PRINT TAB(8,10)"Please wait - cal
culating":count%=400
1190 ENDPROC
1200 :
1210 DEF PROCperscalc
1220 FOR X%=1 TO 20
1230 FOR Y%=1 TO 20
1240 PROCfnz
1250 S=X%-10:Z=Z-5:D=SQR(SQR(Z*Z+S*S)+
Y%*Y%)
1260 P%(X%,Y%,1)=(S/D)*400+600
1270 P%(X%,Y%,2)=(Z/D)*400+800
1280 NEXT Y%,X%
1290 ENDPROC
1300 :
1310 DEF PROCpersdraw
1320 FOR X%=1 TO 9
1330 FOR Y%=20 TO 2 STEP-1
1340 PROCdraw
1350 NEXT Y%,X%
1360 FOR X%=19 TO 10 STEP-1
1370 FOR Y%=20 TO 2 STEP-1
1380 PROCdraw
1390 NEXT Y%,X%
1400 ENDPROC
1410 :
1420 DEF PROCisocalc
1430 FOR X%=1 TO 20
1440 FOR Y%=1 TO 20
1450 PROCfnz
1460 P%(X%,Y%,1)=(X%/2+Y%/2)*60
```





```
1470 P%(X%,Y%,2)=(Z+X%/2-Y%/2)*40+500
1480 NEXT Y%,X%
 1490 ENDPROC
 1500 :
 1510 DEF PROCisodraw
 1520 FOR X%=19 TO 1 STEP-1
 1530 FOR Y%=2 TO 20
 1540 PROCdraw
 1550 NEXT Y%,X%
 1560 ENDPROC
 1570 :
 1580 DEFPROCdraw
 1590 GCOL0,1
 1600 MOVE P%(X%,Y%,1),P%(X%,Y%,2)
 1610 MOVE P%(X%,Y%-1,1),P%(X%,Y%-1,2)
 1620 PLOT85,P%(X%+1,Y%-1,1),P%(X%+1,Y%
-1,2)
 1630 MOVE P%(X%+1,Y%,1),P%(X%+1,Y%,2)
 1640 PLOT85,P%(X%,Y%,1),P%(X%,Y%,2)
 1650 GCOL0,2
 1660 DRAW P%(X%,Y%-1,1),P%(X%,Y%-1,2)
 1670 DRAW P%(X%+1,Y%-1,1),P%(X%+1,Y%-1
,2)
 1680 DRAW P%(X%+1,Y%,1),P%(X%+1,Y%,2)
 1690 DRAW P%(X%,Y%,1),P%(X%,Y%,2)
 1700 ENDPROC
 1710 :
 1720 DEF PROCfnz
 1730 PRINT TAB(12,12)count%:count%=cou
nt%-1
 1740 ON C GOSUB 1780,1790,1800,1810,18
20,1830,1840,1850,1860
 1750 IF P=2 THEN Z=-Z
 1760 ENDPROC
 1770 :
 1780 A=X%-10:B=Y%-10:Z=SQR(ABS(A))*SQR
(ABS(B)):RETURN
 1790 Z=EXP(SIN(X%)*SIN(Y%)*3)/4:RETURN
 1800 Z=EXP(COS(X%/2)*COS(Y%/2)*3):RETU
RN
 1810 A=X%-10:B=Y%-10:T=SQR(A*A+B*B):Z=
COS(T/1.7)*EXP(-T/5)*10:RETURN
 1820 A=X%-10:B=Y%-10:Z=COS(SQR(A*A+B*B
))*2:RETURN
 1830 A=X%-10:B=Y%-10:Z=EXP(6-(SQR(A*A+
B*B)))/35:RETURN
 1840 Z=COS(SQR(X%*X%+Y%*Y%)):RETURN
 1850 Z=EXP(COS(SQR(X%*X%+Y%*Y%))*3)/4:
RETURN
 1860 Z=1:RETURN
 1870 :
 1880 ON ERROR OFF
 1890 MODE 6
 1900 IF ERR<>17 REPORT:PRINT" at line
";ERL
 1910 END
```



HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DOUBLE USAGE - D. Morgan

If you are really concerned with reducing the number of variables used by a procedure, for example when a DIM command is used, don't forget that you can assign a result to a variable used in the calculation. By way of example, consider:

```
DIM N% N%                    A$=RIGHT$(A$,4)+LEFT$(A$,3)
```

These are both quite legitimate Basic statements. The first one will reserve N%+1 (0..N%) bytes, and the second one performs a small amount of string manipulation on A$, and then puts the result back in A$. Note that the variable on the left hand side is not altered until the right hand side has been FULLY evaluated; so A$ in the LEFT$ above still refers to the original A$, and not an intermediate value.

EXTENDING THE MINI TEXT EDITOR

by P. Austin

This article shows how the "Mini Text Editor" published in ELBUG Vol.1 No.9 can be enhanced to provide a number of the facilities available on commercially produced wordprocessors.

This new version of the "Mini Text Editor" has several advantages over the old version. Firstly, and most importantly, it has right justification which means that all your text will appear with a straight edge on the right hand side (like the text in this article). Secondly, it has user defined tabs so that you can design tables in nice neat columns. Other features include variable page length and printer page length (for those people able to use a printer with their Electron). The editor also has page numbering and it can print Basic keywords such as OLD and NEW which the old editor could not do. As an example we have included a piece of text with the respective program lines to show how the editor works (see example 1).

The text editor is in two parts, which completely replace the previous program. The first program is an initialisation routine which places a piece of machine code in the user defined character area and sets up the function keys. The second part is the main Basic program.

SETTING UP THE PROGRAMS

The following procedure should be used to set up the programs:

```
 10 *~LM02~RM35
 20 *~T105~T210
 30 *~T315~T420
100 *Dear Elbug member, \CR
110 *\T2This is the new Mini Text Edi
tor by P.Austin that incorporates a num
ber of features found on expensive word
processors. \CR\CR
120 *\T1It has things like left and r
ight margin settings, right justificati
on, variable page length and line numbe
rs. It also has user defined tabs
130 *which you can use as follows: \C
R\CR
140 *\T1Name \T3Age \T4Score \CR
150 *\T1Alan \T319 \T4100 \CR\T1Dave
\T319 \T437 \CR\T1Geoff \T364 \T417 \CR
\CR\CR
160 *\T1You can also use the variable
s within the text. This is how to use t
hem. \CR\CR
170 *Today is \V1 the \V2 of \V3. \CR
\CR
180 *\T2Yours faithfully, \CR\CR\T3EL
BUG MAG. \CR\CR
```

```
Dear Elbug member,
        This is the new Mini
Text Editor by P.Austin that
incorporates a number of
features found on expensive
wordprocessors.

    It has things like left and
right margin settings, right
justification, variable page
length and line numbers. It also
has user defined tabs which you
can use as follows:

     Name      Age  Score
     Alan      19   100
     Dave      19   37
     Geoff     64   17

    You can also use the
variables within the text. This
is how to use them.

Today is Monday the 29th of
October.

        Yours faithfully,

            ELBUG MAG.
```

1. Type in program 1 and make a security copy. Run program 1 and press Func/f0. Then press Escape and if 'Escape/Edit' appears press Func/f1 and go on to step 2. If this does not happen press Break, load your security copy and carefully check the assembly listing for any mistakes. Then try again.

2. On a fresh cassette save the assembled machine code routine by typing *SAVE"PATCH" B00 D00.

3. Type in program 2 and make a security copy.

USING THE PROGRAM

The program is, in operation, basically similar to the original "Mini Text Editor". The user enters text as Basic lines and then edits them as for a Basic program using the cursor keys, Delete and Copy. The editor now allows the use of two kinds of embedded commands.

1. Commands following a tilde (~):
(a) ~LMnn sets the left hand margin to position nn (based on the absolute left hand margin, which is column 1). The number nn must contain two digits so the value 2, for example, must be entered as 02.

(b) ~RMnn as above but for the right hand margin.

(c) ~PLnn sets the number of lines to be printed before the printer moves onto a new sheet. Again the number must contain 2 digits.

(d) ~SLnn sets the actual number of lines that can fit on the sheets of paper you are using (usually 66). Again the number must contain 2 digits.

(e) ~PNnn sets the initial page number which is incremented on advancing to each new page. If nn is 00 then no page numbers are printed. Again the number must contain 2 digits.

(f) ~Txnn this sets the column to which tabulation x moves to. There are 9 user tabs numbered 1 to 9. To define tab 6 to move to column 20 you would type in ~T620. Again the nn part must contain 2 digits.

Initially LM=01, RM=80, PL=62, SL=66, PN=00 (no page numbers), and all tabs are set to 0.

2. Commands following a backslash (\):

(a) \Tx tabs to the position previously defined by ~Tx. Note that it is impossible to tab backwards.

(b) \Vx is variable x. There are 9 user variables numbered 1 to 9. At print time it is possible to enter a string which will be substituted for a variable. For example, if you enter 'Electron' as variable 1 then whenever the program encounters \V1 in your text it will substitute Electron in its place. This allows several copies of the same document (or letter) to be produced, but each tailored to individual requirements.

(c) \CR produces a carriage return in the print out.

(d) \FF produces a form feed.

Some of the backslash commands can be easily entered by pressing certain keys:
Ctrl/V gives \V (this will not offer the variable number).
Return gives \CR
Ctrl/Return gives a Return followed by a *.
Ctrl/F gives \FF
Thus Ctrl V will enter \V at the current position.

Func/f0 sets up the keys to give the above functions.
Func/f1 restores all the keys to their normal functions.

An asterisk should always be placed at the beginning of each line to stop tokenisation (hence the Ctrl/Return idea). Text line numbers should not exceed 9999.

Key f9 starts the printing routine. The other function keys uses should be self-explanatory from the listing of program 1.

The print routine works as follows:

a) Press Func/f9.
b) Enter the number of copies you require.
c) Enter the number of variables you have in the text. I.e. if you had used variables 1 to 5 you would enter 5.
d) Enter a string for each variable in each copy as prompted.
e) Specify whether or not you require a hard (printed) copy.

NOTE

The print routine considers spaces as word separators. This function is not performed by embedded commands. You must separate the text and a command by using a space (e.g 'on the Electron. \CR').

If you wish to start a new paragraph use \CR\CR. To save any text that you have written, just type SAVE "filename" and press Return. This will always save the main program with your text so that when you wish to re-use the same piece of text you need only LOAD the text and *LOAD the patch program.

To re-use the editor type:
*LOAD PATCH <Return>
LOAD "filename"
and then you are ready to use the editor.

PROGRAM 1

This program sets up both the function keys, to perform various tasks, and a machine code routine which intercepts the 'get character routine' (OSRDCH). The routine checks the character typed at the keyboard to see if it is one of a set of control characters which perform special tasks (explained later). If it is one of these codes the routine places a series of characters in the keyboard buffer by using OSBYTE with A=138 (see User Guide page 283). Once assembled, this routine and the function key definitions are loaded directly into memory, hence wasting no space which can be used for text.

```
  10 REM PROGRAM MINI TEXT EDITOR
  20 REM VERSION E0.2
  30 REM AUTHOR  P.AUSTIN
  40 REM ELBUG   DECEMBER 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 *KEY 0 ?&210=0:?&211=&C|M
 110 *KEY 1 ?&210=80:?&211=220|!|P
 120 *KEY 2 AUTO10|!|P
 130 *KEY 3 AUTO100|!|P
 140 *KEY 4 LIST10,99|!|P
 150 *KEY 5 LIST100,9999|!|P
 160 *KEY 6 DELETE10,99
 170 *KEY 7 DELETE100,9999
 180 *KEY 9 CLEAR:DIMT%(9),V$(9,10):PR
OCPR|M
 190 read=?&210+?&211*256
 200 FOR I=0 TO 1
 210 P%=&C00
 220 [OPT3*I
 230 PHP:STX&70:STY&71:JSR read
 240 CMP#27:BEQ error:CMP#22:BEQ var:C
MP#6:BEQ form:CMP#13:BEQ newl
 250 JMP exit1
 260 .error JMP error1
 270 .var LDY #ASC"V":JSR char:JMP exit
 280 .form LDY #ASC"F":JSR char:LDY #A
SC"F":JSR char:JMP exit
 290 .newl LDA #129:LDY #255:LDX #254:
JSR &FFF4:CPX #255:BEQ end1:LDY #ASC"C"
:JSR char:LDY #ASC"R":JSR char:JMP exit
 300 .end1 LDY #ASC"*":JSR char:LDA #1
3:JMP exit1
 310 .char LDA #138:LDX #0:JSR &FFF4:R
TS
 320 .exit LDA #ASC"\"
 330 .exit1 LDX&70:LDY&71:PLP
 340 RTS
 350 .error1 LDX&70:LDY&71:PLP
 360 BRK
 370 ]:$P%=CHR$10+CHR$10+"Escape/Exit"
+CHR$0
 380 NEXT
```

PROGRAM 2

This is the main Basic program which deals with the text formatting. The procedures used in this program are outlined below with a brief description of their purpose.

PROCPR is the main procedure which calls all the others and gets the values of various printing parameters from the user.

PROCCP advances along the text stored at line 10 onwards calling various other routines to print it, insert variables, right hand justify etc. The VDU1,12 causes a 'form feed' on the printer.

PROCLM is used to give values to variables used by the program which are defined within the text being printed.

PROCCN implements the embedded control strings (placed in the text via the keyboard buffer by PROGRAM 1).

PROCWD places words taken from the text into the print string L$.

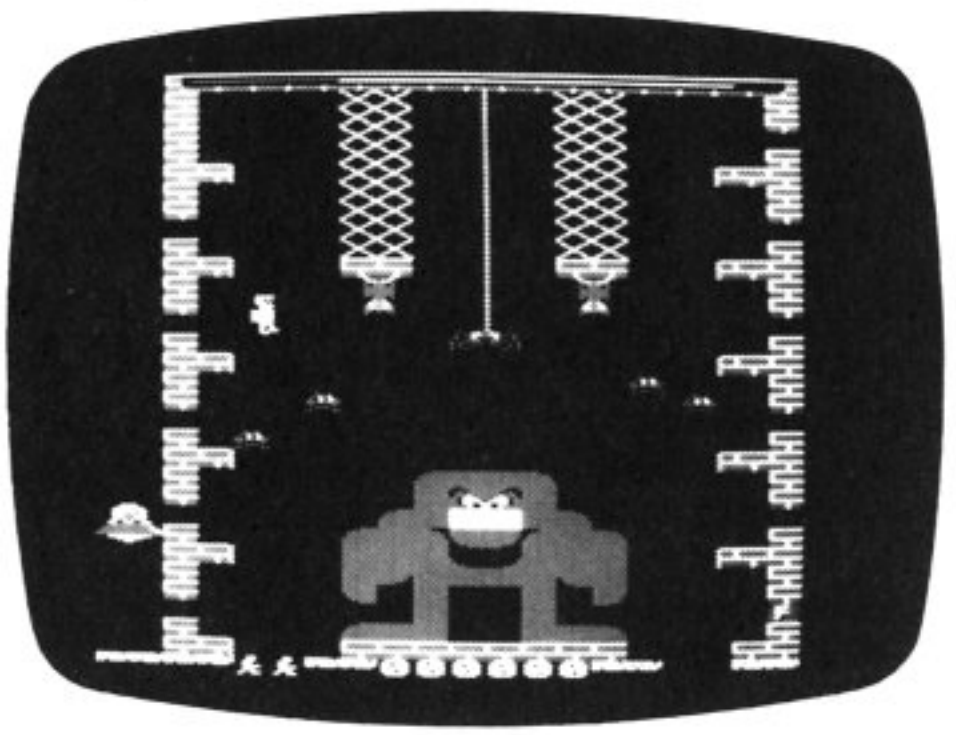
PROCLN does two jobs. Firstly it performs right hand justification by

inserting spaces between words until the line is the right length. Secondly it prints the string L$.

```
10000 * £
10020 LOCAL V%,C%,N%,H%,I%,J%,Q$
10030 PRINT
10040 REPEAT VDU12:INPUTTAB(0,2)"How ma
ny copies ",N%:UNTIL N%>0
10050 PRINT
10060 REPEAT PRINTTAB(19,4)SPC(20):INPU
TTAB(0,4)"How many variables ",V%:UNTIL
 V%>=0 AND V%<=9
10070  IF V%>0 THEN FOR I%=1 TO N%:PRINT
'"In copy ";I%:FOR J%=1 TO V%:PRINT"Vari
able ";J%;" ";:INPUTV$(J%,I%):NEXT:NEXT
10080 PRINT
10090 REPEAT VDU12:INPUTTAB(0,4)"Hard c
opy (Y/N) ",Q$:UNTIL Q$="Y" OR Q$="N":V
DU12
10100 IF Q$="Y" THEN H%=TRUE
10110 IF H% THEN VDU2:*FX6 10
10120 FOR C%=1 TO N%
10130 PROCCP
10140 NEXT
10150 VDU3
10160 ENDPROC
10165 :
10170 DEF PROCCP
10180 LOCAL LM%,RM%,PL%,SL%,P%,Z%,LC%,C
R%,EL%,ET%,L%,PN%,W$,L$
10190 LM%=1:RM%=80:PL%=62:SL%=66
10200 FOR P%=1 TO 9:T%(P%)=P%*8:NEXT
10210 P%=PAGE+4
10220 REPEAT
10230 EL%=FALSE:CR%=FALSE:T%=1:L$=""
10240 REPEAT
10250 LC%=TRUE
10260 P%=P%+1:IF ?P%=13 THEN P%=P%+5
10270 IF ?P%=96 THEN LC%=FALSE:ET%=TRUE
10280 IF ?P%=126 THEN PROCLM:LC%=FALSE
10290 IF ?P%=92 THEN PROCCN:LC%=FALSE
10300 IF ?P%=32 THEN PROCWD:LC%=FALSE:Z
%=P%:W$=""
10310 IF LC% THEN W$=W$+CHR$(?P%)
10320 UNTIL CR% OR EL% OR ET%
10330 PROCLN
10340 L%=L%+1
10350 IF L%>=PL% AND PN%>0 THEN PRINT'S
TRINGS$(LM%+(RM%-LM%) DIV 2-3," ");"PAGE
:";PN%:PN%=PN%+1
10360 IF L%>=PL% AND H% THEN VDU1,12
10370 IF L%>=PL% THEN L%=0
10380 UNTIL ET%
10390 IF PN%>0 THEN REPEAT PRINT:VDU1,1
0:L%=L%+1:UNTIL L%>=PL%:PRINTSTRING$(LM
%+(RM%-LM%) DIV 2-3," ");"PAGE:";PN%
10400 IF H% THEN VDU1,12
10410 ENDPROC
10415 :
10420 DEF PROCLM
10430 LOCAL C$,X%,Y%
10440 C$=CHR$(P%?1)+CHR$(P%?2):X%=VAL(CH
R$(P%?2)):Y%=VAL(CHR$(P%?3)+CHR$(P%?4))
10450 IF C$="LM" AND Y%>0 AND Y%<RM% TH
EN LM%=Y%
10460 IF C$="RM" AND Y%<81 AND Y%>LM% T
HEN RM%=Y%
10470 IF C$="SL" AND Y%>0 THEN SL%=Y%:I
F H% THEN VDU1,27,1,67,1,SL%
10480 IF C$="PL" AND Y%>0 AND Y%<SL%-1
THEN PL%=Y%
10490 IF C$="PN" AND Y%>=0 THEN PN%=Y%
10500 IF LEFT$(C$,1)="T" AND X%>0 AND Y
%>0 AND Y%<80 THEN T%(X%)=Y%
10510 P%=P%+4
10520 ENDPROC
10525 :
10530 DEF PROCCN
10540 LOCAL C$,X%
10550 C$=CHR$(P%?1)+CHR$(P%?2):X%=VAL(C
HR$(P%?2))
10560 IF C$="CR" THEN CR%=TRUE
10570 IF C$="FF" AND PN%>0 THEN REPEAT
PRINT:VDU1,10:L%=L%+1:UNTIL L%>=PL%:PRI
NTSTRING$(LM%+(RM%-LM%) DIV 2-3," ");"P
AGE:";PN%:PN%=PN%+1
10580 IF C$="FF" AND H% THEN VDU1,12
10590 IF C$="FF" THEN L%=0
10600 IF C$="ST" THEN BT%=FALSE
10610 IF LEFT$(C$,1)="V" AND X%>0 THEN
W$=W$+V$(X%,C%)
10630 IF LEFT$(C$,1)="T" AND X%>0 THEN
T%=T%(X%):L$=LEFT$(L$+STRING$(80," "),T
%)
10640 P%=P%+2
10650 ENDPROC
10655 :
10660 DEF PROCWD
10670 IF LEN(L$)+LEN(W$)+1>RM%-LM% THEN
 P%=Z%:EL%=TRUE:ENDPROC
10680 L$=L$+W$+" "
10690 ENDPROC
10695 :
10700 DEF PROCLN
10710 LOCAL I%,S%
10720 IF CR% OR ET% GOTO 10780
10730 I%=T%:S%=LEN(L$)
10740 REPEAT
10750 I%=I%+1:IF I%=S% THEN I%=T%+1
10760 IF MID$(L$,I%,1)=" " AND S%<RM%-L
M% THEN L$=LEFT$(L$,I%)+MID$(L$,I%):I%=
I%+1:S%=S%+1
10770 UNTIL S%=RM%-LM%
10780 PRINTSTRING$(LM%-1," ");L$
10800 ENDPROC
```

THE LATEST GAMES REVIEWED

Title : Jet Power Jack
Supplier : Micro Power
Price : £6.95
Reviewer : Geoff Bains
Rating : ****



Jack has the unfortunate habit of being trapped in a series of 'space garages'. There he has to collect fuel pods from one side of the garage and deliver them to the waiting space ships at the other side.

To help him out in his job, Jack has a jet power pack on his back (okay, so jets don't work in space, but Rocket Power Jack doesn't sound as good) with which he hops around the cavernous garage. You control his left and right movement and the jet thrust.

All this would be easy were it not for the variety of nasty things the owners of the garage leave lying around in the middle.

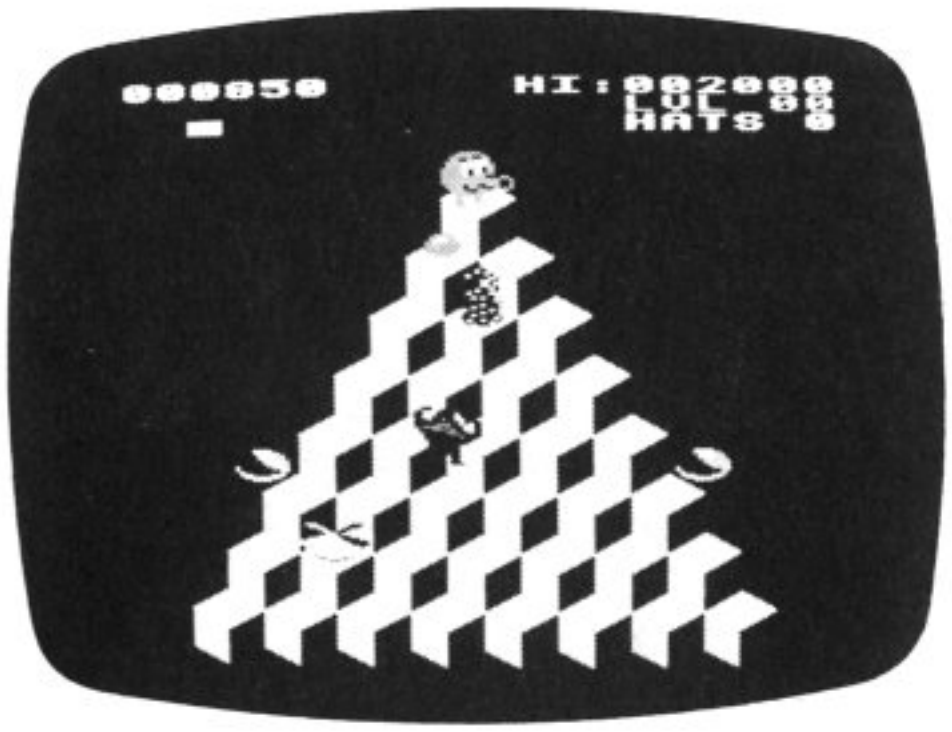
There are five screens to Jet Power Jack, featuring a variety of garages. Some have a series of levels that Jack can use as stepping stones in his quest and some have large areas of empty space with only the odd monster sitting in an inconvenient position in the middle.

The game does not have the most thrilling of plots but once you're into it, you're stuck. It is very frustrating, not all that easy, and totally captivating. As such, it is well worth the money. The graphics are good, the movement smooth and quick, and the controls responsive. Great.

Title : ER*BERT
Supplier : Microbyte
Price : £4.95
Reviewer : Geoff Bains
Rating : ***

ER*BERT is a version of the arcade game, Q*BERT. ER*BERT lives in a giants' causeway of cubes stacked up into various shapes. His aim in life is to turn the cubes to another colour. This he does by hopping onto each of them once or twice. Of course he musn't jump off the edge of the stack or he'll meet a grisly death.

Chasing ER*BERT around his domain are bouncing balls, a gorilla and a snake. Contact with these is also



deadly to ER*B. The gorilla gets particularly nasty if you hop on his banana that lies around. The trick is to lure them to their death by taunting them on the edge of the cube stack. All the other gimmicks are included in this Electron version: helicopter hats for those last minute getaways, rotating transporter discs, and so on.

This version of the game is well implemented. There seems to be a lack of response of the control keys which can make you jump poor ER*BERT over the edge when he was meant to do a breathtaking about-turn just before the precipice. However otherwise the graphics and sound are well up to scratch.

My only complaint is that the game itself is not really very interesting. Still, that is largely a matter of opinion. If you like the arcade version (if you can still find it - there seem to be few Q*BERTs around these days which only goes to show others are not all that chuffed with the game either) then you will not be disappointed with ER*BERT.

Title : Eddie Kidd Jump Challenge
Supplier : Martech
Price : £7.95
Reviewer : Kevin Kilmoore
Rating : **

The roar of the motorcycle beneath you, the rush of wind past your head, the cheers of the crowd; all these things Eddie Kidd can savour when he performs his death-defying stunts. Unfortunately you can't quite get that same atmosphere sitting in front of a TV set frantically pushing keys on your Electron.

In the Eddie Kidd Jump Challenge you have to jump Eddie over a variety of obstacles. You start with a BMX bike over a few barrels and progress on from there to a motorbike that can leap buses.

The cassette insert claims that this game will not only capture the excitement of a real stunt but test your skill at such activities too. Well, I've never jumped my bike over any cars, but even taking the run up is not too true to life in my judgement. Does Eddie Kidd really use an automatic? What happened to the clutch?

Aside from this lack of realism, the Eddie Kidd Jump Challenge is still good fun. The controls (nine keys for the motorbike) take some mastering but once done, sailing over the cars is great fun. The Electron struggles at times to maintain the speed needed for the complex graphics and this version is not as impressive as those for the BBC micro and CBM 64.

Why the 'challenge'? Martech are offering a prize of a TV, a micro, or, more suitably, a BMX bike for the best jumps made in two, two monthly periods. I've a suspicion that even if Eddie was eligible for entry, he wouldn't win.

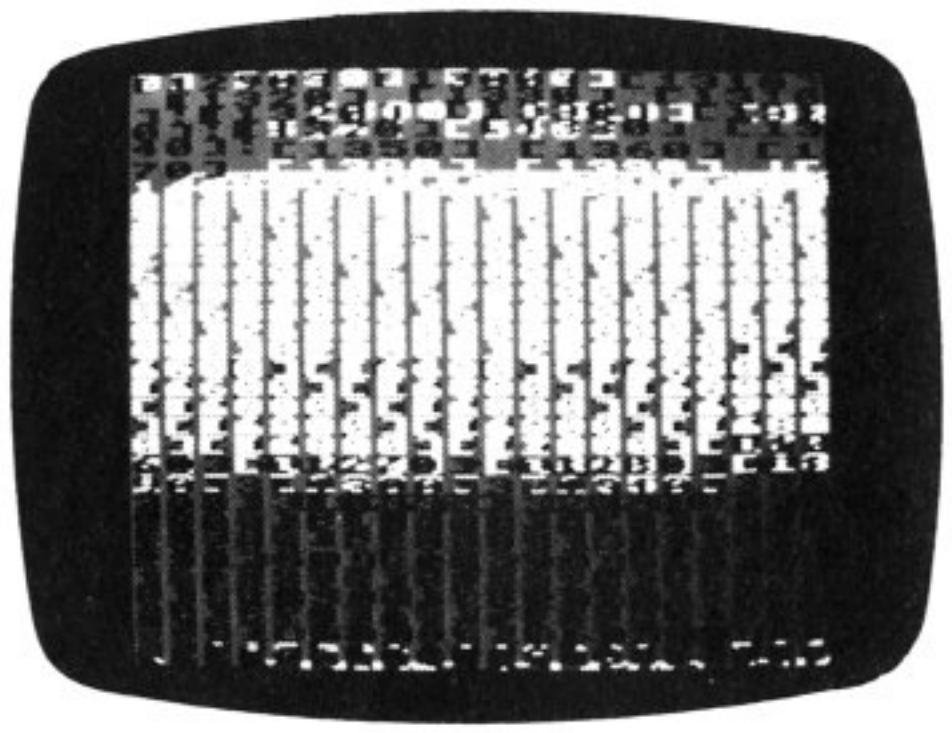
IMPROVED TRACE FACILITY

An aid to debugging programs

by Martin Dale

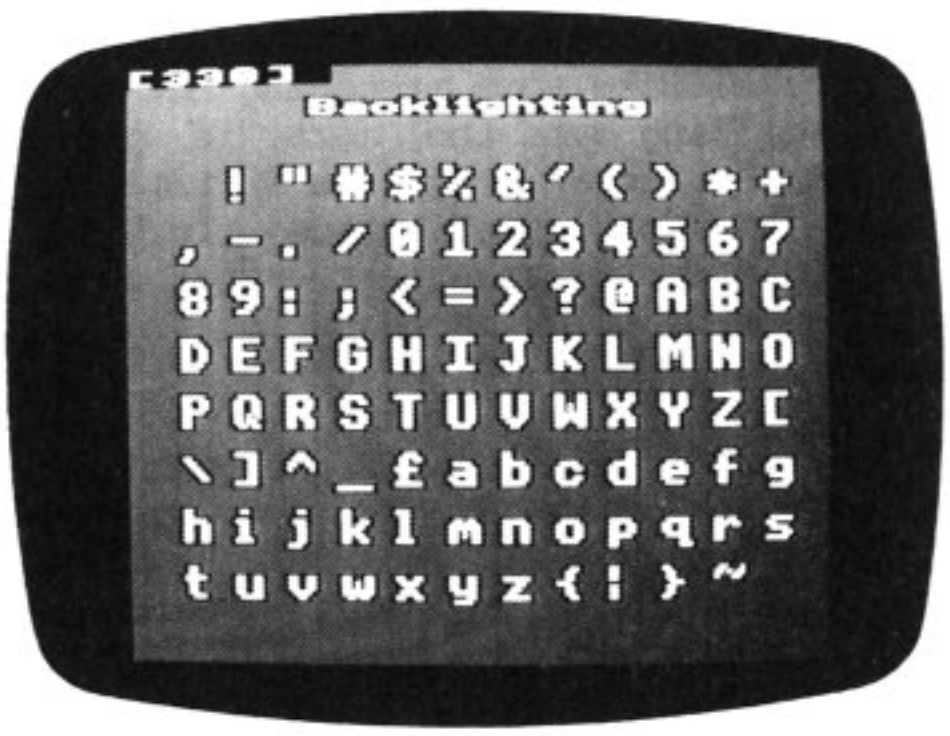
This month we present a useful utility that extends the program debugging 'trace' facility. This routine, entirely written in machine code, will make your Basic programs much easier to get working.

Your Electron has a powerful degugging tool in the shape of the built in trace facility. If you are not already familiar with this facit of Basic you would do well to read page 191 of the User Guide and experiment a little with it. In a nutshell, the command TRACE ON switches on the Basic trace. This causes each program line number to be printed on the screen at the current cursor position as the program line is executed, so that you can follow the progress of your program as it is running. The trace can be switched on in immediate mode or from within a program. It can be turned off using TRACE OFF.



The trouble with the built-in trace facility is that it has a tendency to fill your screen with line numbers, totally obliterating whatever the program is producing. You can see where you are in the program but you can't see what the program is doing. Try running a program like the 3D Surfaces program in this issue, with the trace on, to see this effect. This machine code routine inproves the trace facility to make it much more useful.

There are two basic improvements implemented in this extension: the line numbers are printed only in the top left hand corner of the screen (or current text window), and you have the facility to pause between each line to allow you time to study the current position in the program.



To use the program, type it in very carefully and then save it to tape in case there are any errors in the machine code. Once run, the extended trace facility will initialise itself, and you may then use it as you wish. Note that it will be lost on pressing Break, and that it shouldn't be initialised twice or the machine will hang. As with the standard Basic trace, you use TRACE ON to enable it, and TRACE OFF to disable it.

HOW IT WORKS

The utility works by intercepting the operating system routine OSWRCH, which writes a character, and performing three tests to check for the start of printing of the current line number. These tests are: is the trace flag set, is this character not within a VDU sequence, and is it the "[" character (the trace facility prints line numbers within these square brackets)? If all of these tests are true, then the extended trace routines are called.

The current cursor position is first stored away, as are flags indicating whether the text and graphics cursors are joined and that the trace routine is active. Next, the cursor is positioned at the top left of the screen (more accurately, the current text window). Once this is done, the routine exits back to Basic, allowing the normal trace routines to print the current line number.

When the printing of the current line number is complete, the trace routine is again entered (this is triggered by the trailing space after the line number) and a delay is started. The simple loop will normally wait for 2 seconds, but can be made to terminate quicker by the press of a key, or immediately by a press of the Shift key. This is designed to allow the program to be stepped through at a variety of speeds, allowing correct sections to be passed over very quickly whilst up to 2 seconds can be spent on any lines which seem worthy of more detailed study.

Once the delay has finished, the cursor is restored to its original position, the trace active flag is cleared, and the text and graphic cursor are re-joined if necessary. The routine then exits, and the user's program carries on running as normal.

KNOWN LIMITATION

There is a known limitation in the current version of this program: any attempt to print a "[" as a single character (ie not as part of a graphics sequence) whilst tracing will cause the program to respond un-predictably. There is no easy way to cure this, but knowing that it exists should help you to avoid too many problems.

```
 10 REM PROGRAM TRACER
 20 REM VERSION E0.01
 30 REM AUTHOR  MARTIN DALE
 40 REM ELBUG   JAN/FEB 1985
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 osbyte=&FFF4
110 FOR pass=0 TO 2 STEP 2
120 P%=&0D00
130 [OPT pass
140 .newvec
150 PHA
160 LDA &20          \ Get trace flag
170 BNE trcon        \ Branch if <> 0
180 PLA
190 JMP printchar
200 .exit
210 PLA
220 .exit2
230 LDX temp         \ Restore
240 LDY temp+1       \ registers
250 .printchar
260 JMP (oldvec)     \ Print character
270 .trcon
280 STX temp         \ Save registers
290 STY temp+1
300 LDA #&DA         \ Is it in the
310 LDX #0           \ middle of a
320 LDY #&FF         \ VDU command?
330 JSR osbyte
340 TXA
350 BNE exit         \ Return if yes
360 LDA trcflg       \ In trace output?
370 BNE midtrc       \ Yes, branch
380 PLA
390 CMP #ASC("[")    \ Start of trace?
400 BNE exit2        \ No, return
410 INC trcflg       \ Set trace flag
420 LDA #&86         \ Save cursor
430 JSR osbyte       \ position
440 STY vpos
450 STX pos
460 LDA #&75         \ Is VDU 5 set?
470 JSR osbyte
480 TXA
490 AND #32
500 BEQ ntgraphic    \ No, branch
510 INC trcflg+1     \ Yes, set flag
520 LDA #4           \ VDU 4
530 JSR printchar
540 .ntgraphic
550 LDA #30          \ Move cursor to
560 JSR printchar    \ top left corner
570 LDA #ASC("[")    \ Print left "["
580 BNE printchar    \ and return
590 .midtrc
600 PLA
610 CMP #ASC(" ")    \ Finished ?
620 BNE exit2        \ No, return
630 JSR printchar    \ Clear "]"
640 DEC trcflg       \ Restore flag
650 LDA #31          \ Restore cursor
660 JSR printchar
670 LDA pos
680 JSR printchar
690 LDA vpos
700 JSR printchar
710 LDA trcflg+1     \ Cursors joined?
720 BEQ ntgrphic     \ No, branch
```

```
730 LDA #5          \ Rejoin cursors
740 JSR printchar
750 DEC trcflg+1    \ Clear VDU 5 flag
760 .ntgrphic
770 LDA #21         \ Flush keyboard
780 LDX #0          \ buffer
790 JSR osbyte
800 LDA #200        \ 200 ticks
810 STA delay       \
820 .loopdelay      \
830 LDA #129        \ Shift key
840 LDX #255        \ pressed?
850 LDY #255        \
860 JSR osbyte      \
870 CPY #255        \
880 BEQ exitdelay   \ Yes, exit
890 LDA #129        \ No, wait a
900 LDX #1          \ tick. Key
910 LDY #0          \ pressed whilst
920 JSR osbyte      \ waiting?
930 BCC exitdelay   \ Yes, exit
940 DEC delay       \ No, any more
950 BNE loopdelay   \ ticks?
960 .exitdelay      \ exit delay
970 RTS
980 .delay          \ delay byte
990 NOP
1000 ]
1010 trcflg=P%
1020 temp=P%+2
1030 pos=P%+4:vpos=P%+5
1040 oldvec=P%+6
1050 NEXT
1060 !P%=0:!(P%+4)=0
1070 ?oldvec=?&20E:?(oldvec+1)=?&20F
1080 ?&20E=newvec MOD 256
1090 ?&20F=newvec DIV 256
1100 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

WHICH DAY IS IT?

The short function below allows the day of the week to be calculated from a date entered. Note that the format required by the code below is fairly exacting; the format being DD:MM:YYYY, where DD is the two digit date (with a zero preceding if necessary), MM likewise the month and YYYY the year. The ':' may be replaced by most characters, but it is best to standardise on just one separator.

```
  100 DIMday$(7):FORA%=0TO7:READday$(A%):NEXT:CLS:PRINT
  200 REPEAT
  300 INPUT LINE"date:"A$:D%=VALA$:M%=VALRIGHT$(A$,7):Y%=VALRIGHT$(A$,4)-1600
  400 PRINTday$(FNday)"DAY"
  500 UNTIL0
 2000 DATA NO ,SUN,MON,TUES,WEDNES,THURS,FRI,SATUR
 3000 DEFFNday:L%=(Y%MOD4ORY%MOD100=0ANDY%MOD400)>0:y%=Y%-1:m%=M%-1:=((Y%+y%DIV4-y%
DIV100+y%DIV400+31*m%-(m%DIV2)-(m%=8)-(m%=10)+(m%>1)+(L%ANDm%>1)+D%+6)MOD7)+1AND(D%
>0ANDD%<(32-(m%MOD7MOD2)+(m%=1)+(L%ANDm%=1))ANDM%>0ANDM%<13ANDY%>0)
```

ANOTHER BUG IN BASIC? - C.T.Marshall

Users of the advanced maths functions of Basic may be interested to know of the presence of a bug in the X^Y function. Attempts to use this function (seemingly quite legitimately) will result in a 'Log range' error if Y lies between (but not including) zero and one and X is zero (but not otherwise). Errors can be avoided if a test is made to see if X is 0 before attempting this operation. SQR(0) functions quite correctly though (this being equivalent to 0^.5).

ANOTHER ODDITY IN BASIC - D. Morgan

It is quite well known that Basic does not allow you to use a variable before it has been defined, but there is an exception to this rule; if you define a variable in terms of itself, when it is not already defined, then it will equate itself to zero in order to evaluate the expression in question. For example:

```
A=A+3
```

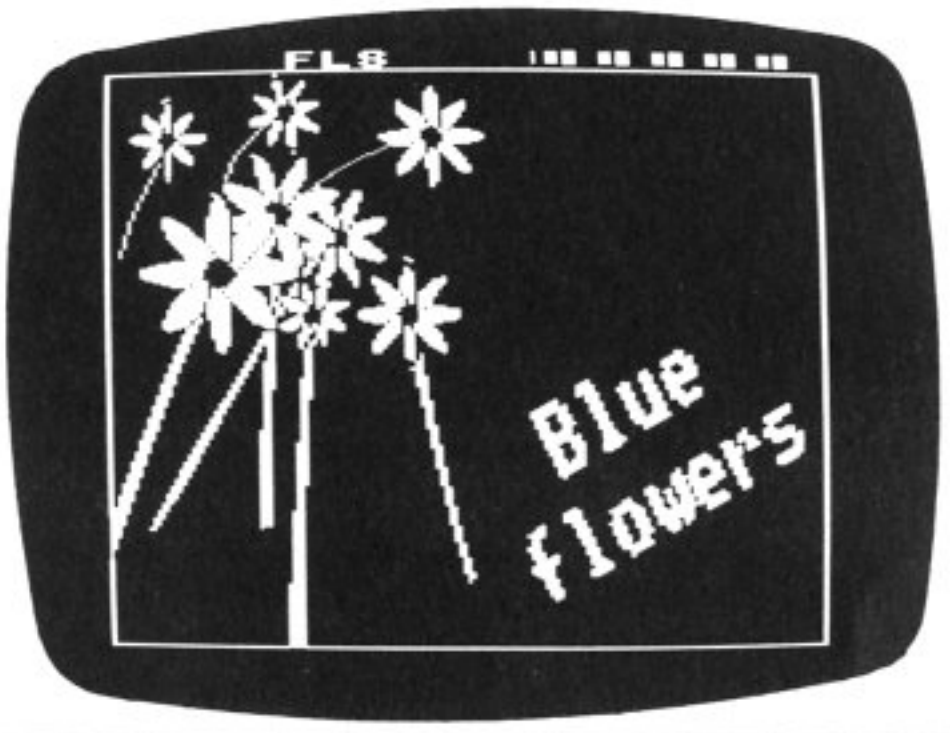
will cause A to be set equal to 3, even if A has not previously been assigned a value.

TWO ELECTRON GRAPHICS PACKAGES

Reviewed by Mike Siggins

Product : Electronic Colouring Book.
Price : £9.95 (inc. VAT)
Supplier: Addison-Wesley Software,
Finchampstead Road,
Wokingham,
Berkshire RG11 2NZ

Product : Picture Maker.
Price : £9.95 (inc. VAT)
Supplier: Acornsoft,
4a Market Hill,
Cambridge, CB2 3NJ



These two recent releases both tackle graphics tasks on the Electron but are very different in their application and scope.

ELECTRONIC COLOURING BOOK

The Electronic Colouring Book is a clever piece of programming, enabling the user to colour in outline pictures using a palette of thirty-five colours, achieved by combining existing colours. The program loads well and quickly, and presents a mode 2 screen with the palette displayed on the right hand side. The background can be changed between either black or white. Cursor movement can be effected by either keyboard commands or by joystick, either one offering good control.

The user is now free to select a picture for colouring. There are some eighteen pictures supplied on the tape, four of which are duplicated in both blank and painted versions. The pictures are of a good standard considering the limitations of mode 2 resolution. It is possible to create your own pictures and save these for colouring but the drawing routine is limited for anything more complex than straight lines. I suspect that the supplied pictures were created on a graphics pad rather than with the program's routines.

Having selected and loaded a picture, the meat of the program is in the painting. This is nicely done with a colour being chosen by 'dipping' the cursor into the palette and then moving it to the section to be filled. The filling is reasonably quick and accurate. One also has the option to correct a painted section by re-filling in the background colour. The completed picture can be saved to tape for later display. A nice option is the facility to change the screen palette by using the number keys. By entering values, the palette is cycled by use of the VDU 19 command to give even more shades. This is best carried out slowly as a picture can get in quite a mess, though default colours are restored easily.

The documentation supplied is brief but seems to cover all the points adequately. It also includes representations of the pictures supplied on the cassette.

The program does have some faults as it stands. It is possible to clear the screen at any time using the command 'C', but this is accepted without any

kind of check. The proximity to the frequently used 'D', for delete, aggravates the problem. Another problem is the colour shades themselves. The colours are mixed in such a way as to cause 'waves' on a domestic television. The results are excellent on a colour monitor but for the majority of users, the screen display will be of a lower standard.

PICTURE MAKER

The Picture Maker program from Acornsoft is entirely different in scope. It is a utility which enables the creation of complex screen displays. The pictures and designs possible are very impressive as proved by the demonstration files supplied. It is supplied in the usual Acorn card box with a comprehensive manual.

The Picture Maker operates at two levels, Catalogue and Picture. The former takes care of housekeeping tasks including filing, naming, free memory and asterisk commands. The latter is concerned with actual creation of the 'units' around which the program is based. A unit can be anything from a single line up to a full screen picture. The program enables the user to draw and then manipulate units into other pictures, as each unit can be saved in its own right. This removes the need present in most graphics packages to save pictures as a whole. Now, the user can select elements as required from cassette and combine them to create new designs.

The Picture level includes a wide range of facilities including single points, full and dotted lines, triangles, arcs, sectors, circles, fill and text. These are all well implemented with the possibility to correct any mistakes. Although the delete command again fails to ask for confirmation, the effect here is to delete only the last element drawn, not the whole picture. It is only possible to draw in either mode 4 or 5 due to the size of the program. This limits the colours provided for drawing but by using another supplied program, described later, all sixteen colours are available.

Shapes can be drawn and then filled with relative ease. Some very impressive shapes can be built up and then shrunk, expanded and distorted at will. The potential is enormous, limited only by the creative skills of the user. Text can be drawn at any size, angle or shape and slanted to give italic characters. The standard of text is not brilliant, as the larger sizes tend to emphasize the jagged edges of the normal fonts. Nevertheless, the scope is there.

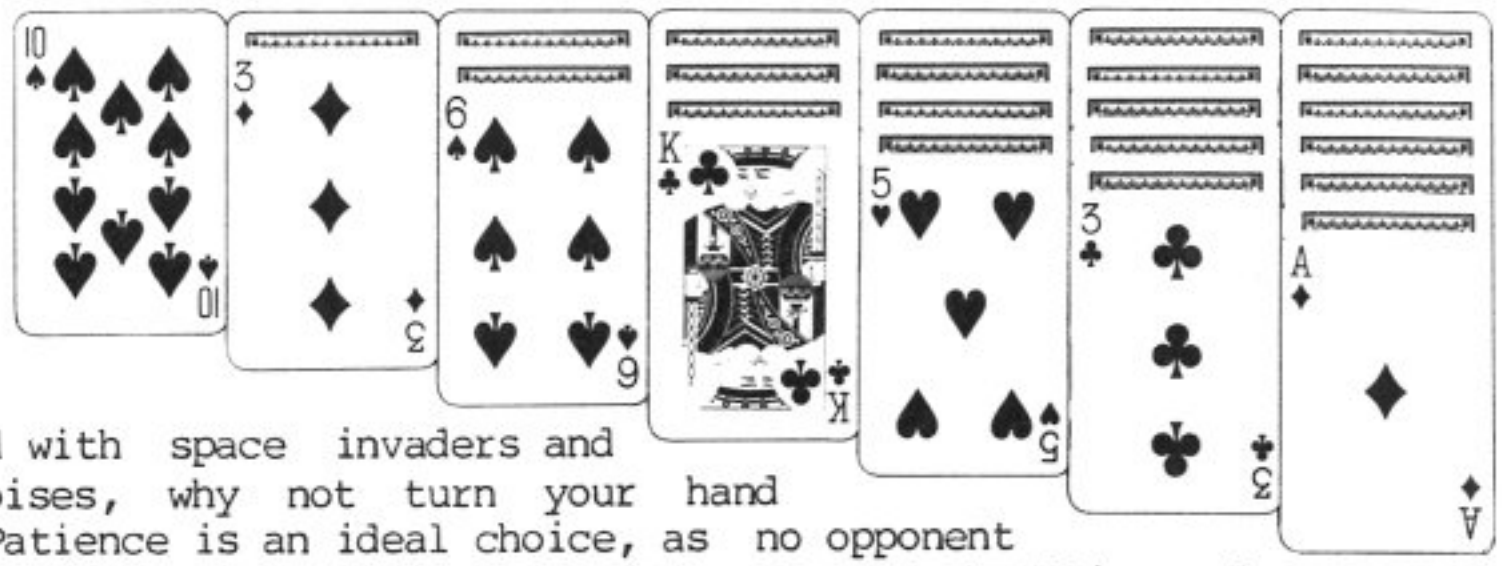
Picture Maker does not stop at creating units and pictures. Further programs are supplied to enable the units to be displayed in different modes and colours and to use the units in your own programs. The Showpic program enables a picture saved from Drawpic to be displayed in higher resolution or with more colours. Full instructions are provided for this, the only restriction being that large units from Drawpic may not fit into the 20k graphic modes. Datapic and Picdata deal with transferring pictures into data for incorporating in basic programs and then producing pictures from such data.

The documentation is very well written and gives a comprehensive guide to all the facilities available. It also has a clear tutorial section for each feature which makes this package, which is complex in places, easy to learn and use.

The two programs perform very different functions. The Colouring Book is well done but, I feel, will not sustain the continued interest that Picture Maker provides. That said, many users, especially younger ones, will find it good fun. The colours are very impressive and the program works well. Picture Maker is a fine piece of work, providing some very complex and sophisticated features with the minimum of fuss. This program is certainly worth the money and is recommended.

PATIENCE

by M. C. Ironmonger



When you get bored with space invaders and constant zapping noises, why not turn your hand to playing cards. Patience is an ideal choice, as no opponent is required, and the computer takes all the effort out of shuffling and dealing the cards.

In this popular version of patience, you try to make four complete piles of cards (one of each suit starting with the ace and ending with the king).

The Electron will deal the cards in the usual formation. This consists of seven columns of cards being dealt face down; the first column will have one card, the second will have two cards etc, all the way up to the seventh column which has seven cards. The last card in each column will be dealt face upwards.

You are then given the rest of the pack turned face down (called the stack), and you turn the cards over three at a time. Initially, most of the play takes place with the seven columns of cards where the object is to build up alternating red/black sequences of cards in descending order. Thus a five of hearts from one column may be placed on a six of clubs in another, and the five and six together could then be placed, say, on a seven of diamonds. All the face up cards in any column are moved together determined by the highest value card in that set. If you have an empty column, then the only card (or set of cards) you are allowed to place there is one where the king is the highest card.

If you turn up an ace, you should place this on the appropriate symbol at the bottom right of the screen, and then start to build up these piles of cards in suits in the usual card sequence (ace, 2, 3, 4 etc). Cards can be moved to these piles directly from the stack or from the bottom card of any of the seven columns above.

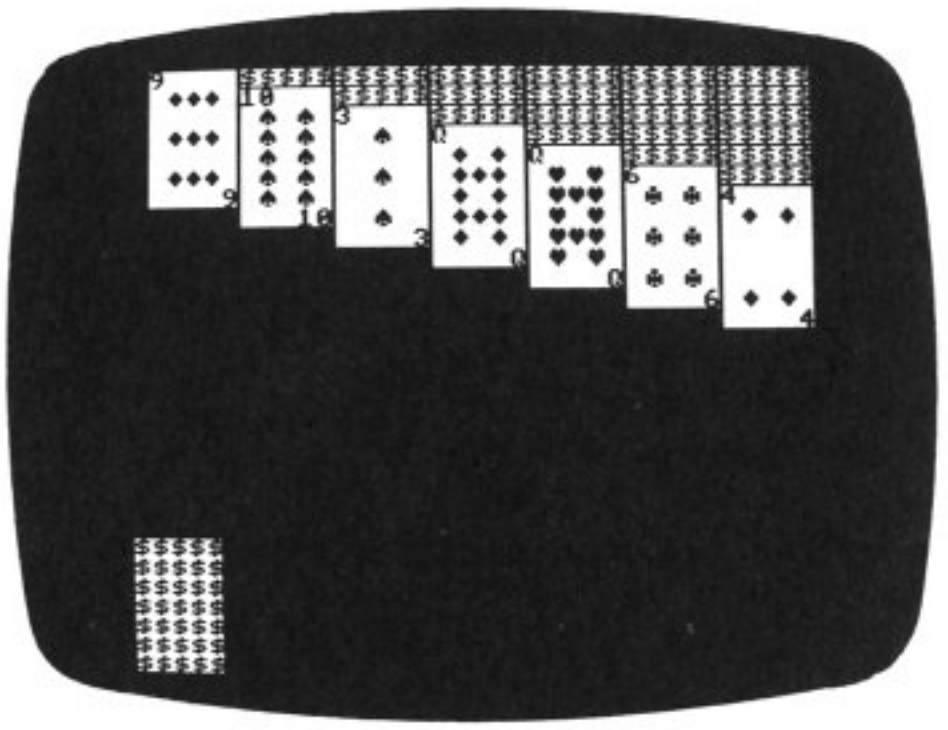
To move cards around, you must use the J and K keys which move the arrow pointer displayed on the screen. First move the arrow to the card you wish to move, then press the spacebar. Once that is done you must then move the arrow above the destination pile or column for that card and then press the spacebar again.

If you are defeated and fail to complete a game (more than likely), then press the Escape key to start a new game.

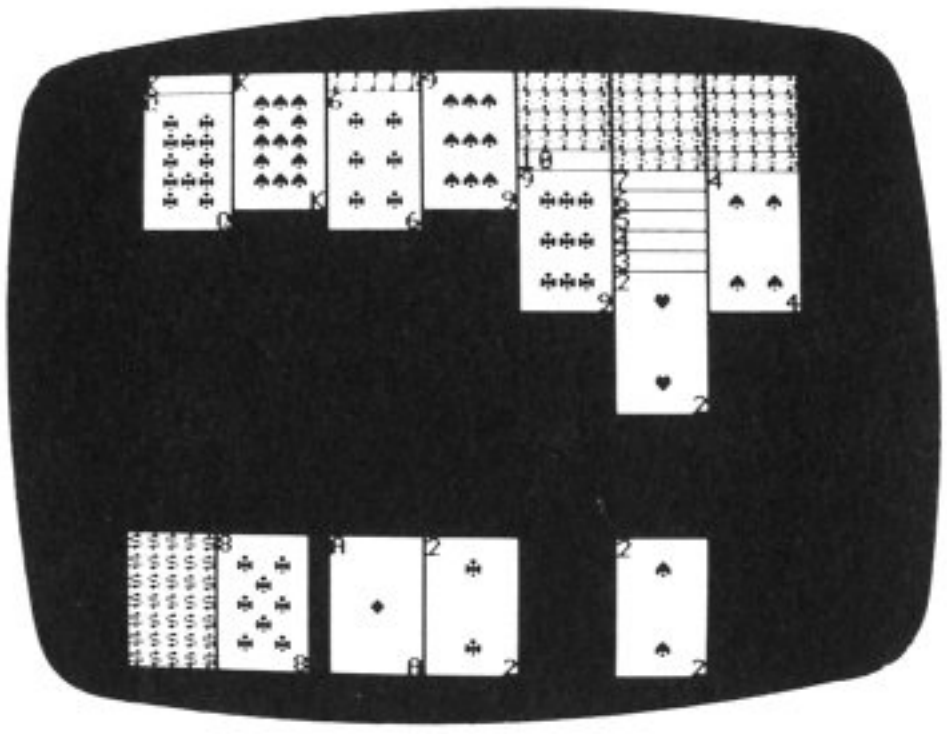
The instructions may sound complicated, but it is suprising how quickly you can learn to play this simple but addictive game.

Although we here at ELBUG strive to publish well written and structured programs, we felt that this program was worth publishing, despite its lack of clarity and good structure, because it was such a good card game. Please take care when typing the program into your Electron as any mistakes will be that much harder to find and correct.

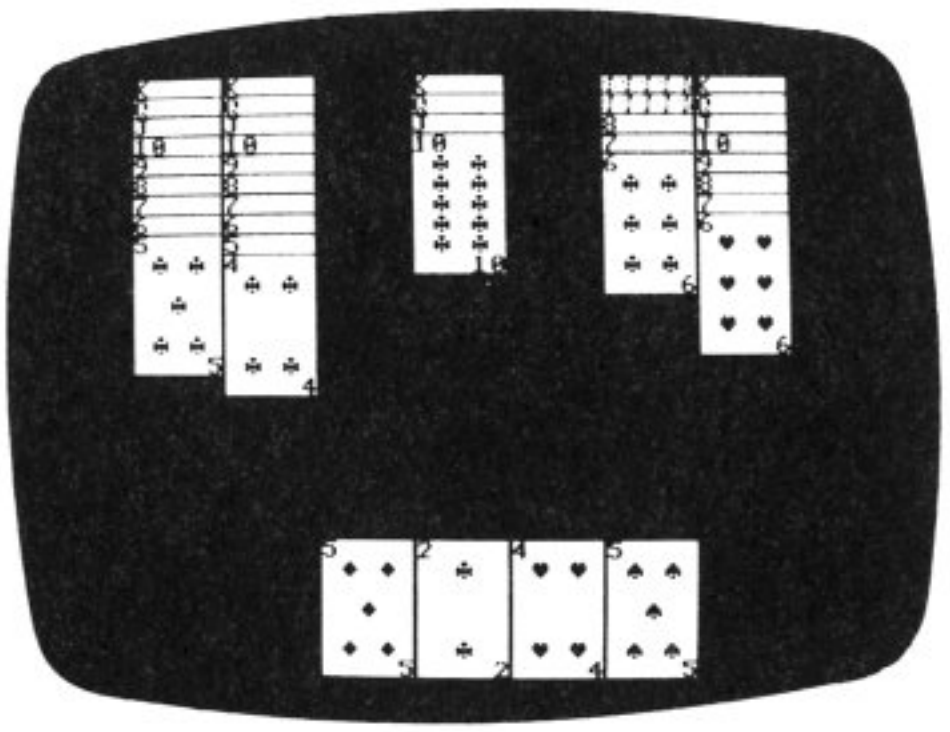
```
 10 REM PROGRAM PATIENCE
 20 REM VERSION E1.4
 30 REM AUTHOR  M.C.Ironmonger
 40 REM ELBUG   JAN/FEB 1985
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 MODE1
110 VDU23,1,0;0;0;0;
120 ON ERROR GOTO 2500
130 DIMA$(12),Z(51),C$(14)
140 PROCcards
150 PROCtitle
```



```
160 PROCshuffle
170 PROCdeal
180 FORZ=1TO4
190 PROCdefine(Z)
200 PRINTTAB(Z*5+9,28)CHR$(255)
210 NEXT
220 H=9997:I=0:J=-1
230 PRINTTAB(4,0)CHR$(254)
240 PROCget
250 I=H MOD13
260 IF A$<>" " THEN 360
270 IF I=7 THEN 1520
280 IF J=-1 THEN PROCfrom ELSE PROCto
290 COLOUR130
300 IF A$(9)+A$(10)+A$(11)+A$(12)<>"2
5385164" THEN 360
310 COLOUR0
320 PRINTTAB(2,5)"Well done, press an
y key to continue"
330 A=GET
340 CLEAR
350 RUN
360 COLOUR0
370 PRINTTAB(I*5+4+36*(I>6)-(I>8),-23
*(I>6))CHR$(254)
380 GOTO240
390 :
1000 DEF PROCfrom
1010 IF I<7 THEN 1070
1020 IF I>8 THEN J=0:GOTO 1570
1030 IF K=0 THEN 1670
1040 IF K=L THEN 1670
1050 J=I:J$="+"+STR$(Z(L)):GOTO 1080
1060 :
1070 J=I:J$=LEFT$(A$(J),INSTR(A$(J),"-
")-1)
1080 ENDPROC
1090 :
1100 DEF PROCto
1110 IF I>8 THEN J=0:GOTO 1570
1120 IF I>6 THEN 1670
1130 A=VAL(MID$(A$(I),2,2))
1140 B=VAL(RIGHT$(J$,2))
1150 IF A=0 AND B MOD13=12 THEN 1170
1160 IF A-B<>14 AND A-B<>-12 AND A-B<>
-38 AND A-B<>40 THEN 1670
1170 D=LEN(A$(I))/3+2
1180 PROCcard(I*5+2,D,VAL(RIGHT$(J$,2)
))
1190 A$(I)=RIGHT$(J$,3)+A$(I)
1200 J$=LEFT$(J$,LEN(J$)-3)
1210 IF J$<>"" THEN 1170
1220 IF J=8 THEN 1340
1230 A$(J)="+"+RIGHT$(A$(J),LEN(A$(J))
-INSTR(A$(J),"-"))
1240 A=LEN(A$(J))/3+8
1250 A=LEN(A$(J))/3+8
1260 IF LEFT$(A$(J),2)="++" THEN A=1:A
$(J)="":GOTO1280
1270 PROCcard(J*5+2,A-7,VAL(LEFT$(A$(J
),3)))
1280 COLOUR130
1290 FOR B=A  TO23
1300 PRINTTAB(J*5+2,B)SPC(5)
1310 NEXT
1320 J=-1
1330 GOTO 1730
1340 J=-1:K=K-1
1350 IF K=0 THEN 1470
1360 IF K<>L THEN 1420
1370 COLOUR130
1380 FOR A=24 TO 31
1390 PRINTTAB(6,A)SPC(5);
1400 NEXT
1410 GOTO 1730
1420 FOR A=L TO K
1430 Z(A)=Z(A+1)
1440 NEXT
1450 J=-1:L=L+3
1460 GOTO 1520
1470 COLOUR130
1480 FOR A=24 TO 31
1490 PRINTTAB(1,A)SPC(10);
1500 NEXT
1510 GOTO 1730
1520 L=L-3
1530 IF L=-3 THEN L=K:GOTO 1520
1540 IF L<0 THEN L=0
1550 PROCcard(6,25,Z(L))
1560 J=-1:GOTO 290
1570 IF VAL(A$(J))=VAL(A$(I))+1 THEN 1
620
1580 J=J+1
1590 IF J<8 THEN 1570
1600 IF VAL(A$(I))+1<>Z(L) OR K=L THEN
1670
1610 A$(J)="+"+STR$(Z(L))
1620 IF I-8<>VAL(A$(J)) DIV 13 THEN 16
70
1630 A$(I)=MID$(A$(J),2,2)
1640 PROCcard(I*5-33,25,VAL(A$(I)))
1650 IF J=8 THEN 1340
```



```
 1660 A$(J)="+"+RIGHT$(A$(J),LEN(A$(J))
-4):GOTO 1250
 1670 PRINTTAB(13,0)"That won't go"
 1680 SOUND1,-9,15,7
 1690 FOR A=1TO1500
 1700 NEXT
 1710 PRINTTAB(13,0)SPC(13)
 1720 J=-1
 1730 ENDPROC
 1740 :
 1750 DEF PROCcard(A,B,C)
 1760 COLOUR131
 1770 IF C=0 COLOUR2:GOTO 1800
 1780 PROCdefine(C DIV13)
 1790 C=C MOD 13+1
 1800 PRINTTAB(A,B)C$(C);
 1810 MOVE A*32,1024-B*32
 1820 PLOT3,156,0
 1830 PLOT3,0,-224
 1840 PLOT3,-156,0
 1850 PLOT3,0,224
 1860 ENDPROC
 1870 :
 1880 DEF PROCdefine(z)
 1890 IF z=1 COLOUR1:VDU23,255,8,28,62,
127,62,28,8; ELSE IF z=2 COLOUR0:VDU23,
255,28,28,107,127,127,107,28;
 1900 IF z=3 COLOUR1:VDU23,255,54,127,1
27,127,62,28,8; ELSE IF z=4 COLOUR0:VDU
23,255,8,28,62,127,127,107,28;
 1910 ENDPROC
 1920 :
 1930 DEF PROCcards
 1940 A$=CHR$10+STRING$(5,CHR$8):F$=CHR
$255
 1950 B$=STRING$(5,CHR$32)+A$
 1960 C$=STRING$(2,CHR$32)+F$+STRING$(2
,CHR$32)+A$
 1970 M$=CHR$32
 1980 D$=M$+F$+M$+F$+M$+A$
 1990 E$=M$+F$+F$+F$+M$+A$
 2000 C$(0)=STRING$(6,"$$$$$"+A$)+"$$$$
$"
 2010 G$=STRING$(4,CHR$32)+A$:G2$=STRIN
G$(4,CHR$32)
 2020 H$=B$+C$:I$=D$+B$:J$=D$+G2$:K$=D$
+D$+D$+D$+D$
 2030 C$(1)="A"+G$+B$+H$+B$+B$+G2$+"A"
 2040 C$(2)="2"+G$+C$+B$+B$+H$+G2$+"2"
 2050 C$(3)="3"+G$+C$+H$+H$+G2$+"3"
 2060 C$(4)="4"+G$+I$+B$+B$+J$+"4"
 2070 C$(5)="5"+G$+I$+C$+B$+J$+"5"
 2080 C$(6)="6"+G$+I$+I$+J$+"6"
 2090 C$(7)="7"+G$+D$+C$+I$+J$+"7"
 2100 C$(8)="8"+G$+D$+C$+D$+C$+J$+"8"
 2110 C$(9)="9"+G$+E$+B$+E$+B$+E$+G2$+"
9"
 2120 C$(10)="10"+STRING$(3,CHR$32)+A$+
K$+STRING$(3,CHR$32)+"10"
 2130 C$(11)="J"+G$+D$+D$+E$+D$+J$+"J"
 2140 C$(12)="Q"+G$+D$+E$+D$+E$+J$+"Q"
 2150 C$(13)="K"+G$+E$+D$+E$+D$+E$+G2$+
"K"
 2160 ENDPROC
 2170 :
 2180 DEF PROCtitle
 2190 COLOUR129:PRINTSTRING$(56,CHR$32)
;"PATIENCE";STRING$(56,CHR$32)
 2200 COLOUR128:COLOUR2:PRINTTAB(16,7)"
left - J"TAB(15,9)"right - K";TAB(8,11)
"pick up/drop - SPACE BAR"
 2210 COLOUR130:COLOUR1:PRINTTAB(6,20)"
Press the SPACE BAR to start"
 2220 *FX4 2
 2230 ENDPROC
 2240 :
 2250 DEF PROCshuffle
 2260 FORA=0TO51:Z(A)=A+13:NEXT:FORA=9T
O12:A$(A)=STR$(A*13-105):NEXT
 2270 FORA=0TO51:B=RND(51):C=Z(B):Z(B)=
Z(A):Z(A)=C:NEXT
 2280 REPEATUNTILGET=32:C=51:A=RND(-RND)
 2290 ENDPROC
 2300 :
 2310 DEF PROCdeal
 2320 VDU19,2,2,0,0,0
 2330 VDU23,254,0,24,24,153,219,126,60,
24
 2340 COLOUR130:CLS
 2350 FOR B=0 TO 6
 2360 A$(B)="+"+STR$(Z(C))+A$(B)
 2370 PROCcard(B*5+2,B+2,Z(C))
 2380 C=C-1
 2390 IF B=6 NEXT:GOTO 2450
 2400 FOR A=B+1 TO 6
 2410 PROCcard(A*5+2,B+2,0)
 2420 A$(A)=STR$(-Z(C))+A$(A)
 2430 C=C-1
 2440 NEXT A,B
 2450 K=24:L=24
 2460 PROCcard(1,25,0)
```



```
2470 COLOUR130
2480 ENDPROC
2490 :
2500 ON ERROR OFF:MODE 1:*FX4
2510 ON ERROR GOTO 2500
2520 IF ERR=17 GOTO 2550
2530 REPORT:PRINT" at line ";ERL
2540 END
2550 COLOUR129:PRINTSTRING$(56,CHR$32)
;"PATIENCE";STRING$(56,CHR$32)
2560 COLOUR128:COLOUR2:PRINTTAB(10,7)"
Another Game ?";
2570 REPEAT:G$=GET$
2580 UNTIL G$="N" OR G$="Y"
2590 IF G$="Y" THEN CLEAR:RUN ELSE MOD
E 6:END
2600 :
2610 DEF PROCget
2620 A$=GET$
2630 PRINTTAB(I*5+4+36*(I>6)-(I>8),-23
*(I>6))" "
2640 H=H+(A$="J")-(A$="K")
2650 ENDPROC
```

NEWS NEWS NEWS NEWS

ACORN BITS

Acorn quietly unveiled a whole host of goodies for the Electron at the recent Compec show at Olympia. A prototype of the Electron 'Tube' interface was on display running a Z80 second processor. The tube interface allows the Electron to be connected to any of the three second processors that are available for the BBC micro - Z80, 6502, and 32016 - giving more memory for program storage, faster processing, and generally more power to your elbow. Acorn promises that the Tube interface will be available soon, but there is no price on it as yet.

PLUS-3

The long awaited Plus-3 interface was also there for the playing with at Compec. The Plus-3 offers Electron owners a single 300K, three and a half inch disc drive complete with an 'advanced' disc filing sytem (ADFS). The whole package costs £299.

THE SOFT STUFF

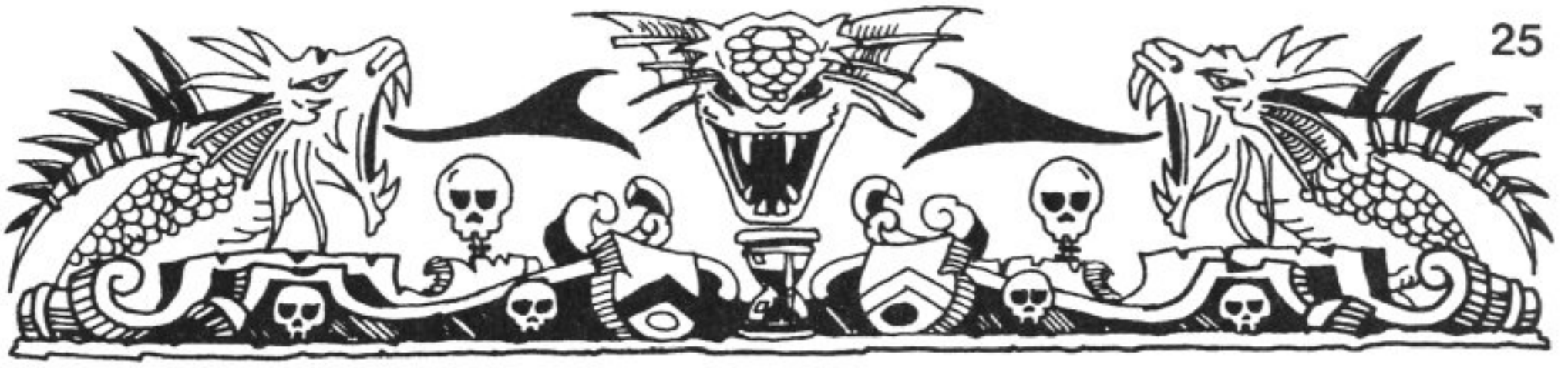
On the software side, too, Acorn had a couple of tricks up its corporate sleeve at Compec. Acornsoft launched its own Logo. The Acornsoft version is, we are told, the definitive version. It comes complete with a utilities cassette and three manuals, all for the sum of £59. The BBC micro ISO Pascal officially received its certificate of worthiness from the British Standards Institution and a version for the Electron was on display. This came in the form of two catridges for the Plus-1. Acorn says that ISO Pascal will be available for the Electron very shortly.

VIDEO

A little off the beaten track, Acorn has launched yet another subsidiary company to 'develop the interactive video market'. Acorn Video has already produced its first product: the Acorn Interactive System. AIS comprises a BBC micro, a Phillips video disc player and connecting electronics. This system allows the storage and replay of over 100,000 video pictures and computer displays in an order determined by the interactive program running on the Beeb. Systems start at about £3000.

MORE AND MORE

Somewhat amazingly, Acorn now claim to have sold a grand total of 170,000 Electrons and to be confident of selling another 200,000 over the Christmas period. Over £2 million is being shelled out in TV advertising to try to realise these hopes.



ADVENTURE GAMES

by 'Mitch'

If you are tired of all those super fast, mega-death, action games, and hanker after quieter, more imaginative challenges to your intellect, then the world of Adventure games may be just what you are seeking. Our own Dungeon Master, 'Mitch', sets the scene in his own inimitable way.

Did you know that there is an old oak door inside the Electron? Should you ever open it and enter you might find your life will never be the same again. Leave your Tee-shirt or pin-striped suit behind. In here, worn leather armour or the cloak of a wizard will be more appropriate. If you've never peeped through before, stay close and I'll show you some of the wonders that lie so close by.

Not so long ago two programmers named Crowthers and Woods had a brilliant idea. What if they wrote a program which could describe a fabulous world of Demons and Magic and then allow the player to explore it by giving simple commands in English. By typing instructions such as 'ENTER BUILDING' or 'OPEN DOOR' the player would be able to tell the computer what he wished to do and where he wished to go. In reply the computer could then describe where the player now stood and what he could see around him. In this way you could decide whether you should 'GO NORTH', towards the dark ruined castle, or 'GO EAST' down the woodland path into the entrance of the evil smelling cave. Whichever way you choose, you can be sure there will be excitement, danger and the chance of matching your wits against all the perils of Adventureland.

As the original adventure games were written on large main-frame computers which had no graphic facilities, but unlimited room for text, this tradition has persisted into the home micro. For this reason mode 6 is the favoured adventure mode on the Electron, as this gives maximum storage for text but no graphics. Not that this is necessarily a bad thing as text only adventures are by and large the best. Pretty picture books are no substitute for a well written adventure story.

There are two main groups of computer fantasy games - 'Adventures' and 'Dungeon & Dragons'. The former is a computer controlled exploration through worlds of imagination, be they in the mists of a magical forest or in the cold, steel corridors of a long abandoned space-ship. In these worlds, the player has the time to map and puzzle his way through the various dangers and perils which are all around him. Dungeon & Dragon games generally involve actual combat between the player and a host of computer-controlled monsters. These contests are a mixture of strategy and quick reactions which can cause untold damage to man and keyboard!

As a relative newcomer, the Electron does not as yet have a large selection of adventure software available to it. Many games are reduced versions of games written for its larger brother, the Beeb. However, the list is growing and already there are some of note.

"SADIM CASTLE" - MP Software, 165 Spital Rd, Bromborough, Merseyside. Price £7.50.

This adventure (reviewed in ELBUG Vol.2 No.2) takes place in the haunted Sadim Castle in which the former lord murdered his wife. Ghosts, vampires and other nasties ooze out of the stonework to prevent you finding the remains of Lady Leonora and laying her to rest.

"WHEEL OF FORTUNE" - Epic Software, 10 Gladstone St, Kibworth Beauchamp, Leicester. Price £9.95.

This game is one of the current top sellers for the Beeb, and a text only version is available for the Electron. The game incorporates all the classic ideas from the past into one game. The program accepts instructions in either the standard two word format (e.g. 'GET BUCKET') or longer involved commands such as, 'ENTER HOUSE THEN PICK UP EVERYTHING AND GO EAST'. In addition, the game features wandering characters who will answer questions and assist you in your quest, should you remember to treat them kindly. The story line involves the recovery of the stolen 'Wheel of Fortune' which has magical powers. To recover it, you must explore a land of caves and magic, and overcome the trolls and dragons which guard the many other treasures, which some careless programmer seems to have dropped everywhere. A note of caution, in this world things tend to disappear if left lying about, so put everything in a safe place before wandering off.

Running home each evening to wield a singing sword in the depths of your latest dungeon, can be a great way to get rid of inhibitions but never forget, the pen is mightier than the sword. Writing adventure games can be more fun than playing them! Basic is too slow for the writing of really good arcade games, but it is perfectly adequate for adventures. Books and magazine articles abound explaining all the necessary techniques to get you started. A word of warning at this point. Before adopting any method, do ensure that it uses procedures and is not a tangled web of GOTO's which the author created long ago on a ZX81 and now, with a minimum of changes, is trying to pass off for the Electron.

One book I can recommend is "How To Write Adventure Games - For the BBC Micro and Electron" by Peter Killworth (Penguin £5.95).

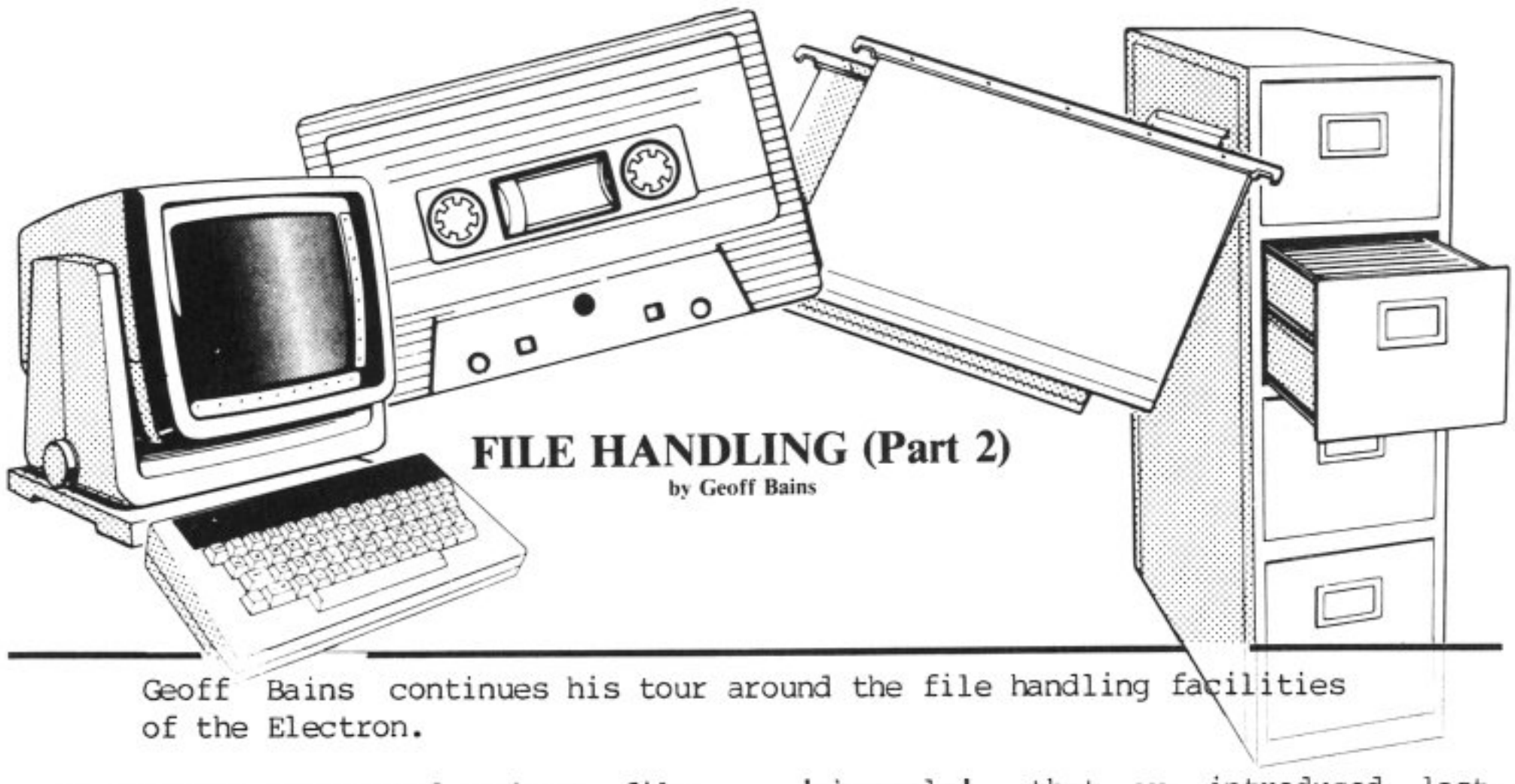
This book is written by the author of the main Acornsoft adventures. The techniques are clearly explained and beautifully structured. D & D plus standard adventure games are explained and many examples are given. Considering the price of most computer related items, this book is a steal and should be required reading for all Acorn adventure writers.

Whilst the inclusion of graphics in Electron adventures is awkward it is possible, as is shown by BUG BYTES' "TWIN KINGDOM VALLEY". As a novice wizard creating your own world, it's probably best to avoid this option as it will eat up too much store. Sound however is a different matter. Including sound effects can spice up any game, so don't forget, while it's easier for you to print "BANG", it's a lot more fun for the player to hear it! Scour the computer mags for sound effects, and store them in your spell book for the day you need the sound of a rusty door or the roar of a dragon.

The most important part of the adventure is the text. A small game with ten carefully laid out, and well described rooms, is worth a hundred boring rooms. Remember that you are trying to create a world that will intrigue, amaze and capture the imagination, not an electronic crossword! Humour is an underated aspect which can turn a simple game, into a winner.

The world you create can be anywhere in time and space, but it's easier to handle if you keep it in an enclosed area (e.g. a cave or castle). It's not logical to tell a player in the middle of a desert that he can only go north. It is also easier, (and more fun) to include magic or supernatural happenings, as this will enable you to have more unusual problems and solutions. Science fiction is also a good source of inspiration - matter transmitters, time-machines and alien monsters. In your world, that innocent public telephone box could turn into a Tardis at the drop of a penny!

Well, I must be off. I've heard of a princess that requires saving , and I know the nearby villagers are terrified of the strange screams that are coming from the local abbey. Why don't you pop an adventure cassette into the slot, grab a stout wooden staff, and go have a peep round that door.



FILE HANDLING (Part 2)

by Geoff Bains

Geoff Bains continues his tour around the file handling facilities of the Electron.

In part one we created a short file on cassette containing the names of eight home computers on the market. Now we want to use that data file. We created the file using the basic sequence

```
OPENOUT
PRINT #
CLOSE #
```

We opened a 'channel' for output with OPENOUT, output data to the file using PRINT# and then closed the file again when we had finished. The procedure to read data into the micro is very similar.

```
OPENIN
INPUT #
CLOSE #
```

The channel is now opened for input. Data is read in with the command INPUT#, and again the file must be closed at the end. Here is a simple program to read back the data stored last time.

```
10 REM cassette file input demo 1
20 CAROL=OPENIN "MICROS"
30 FOR I%=1 TO 8
40 INPUT #CAROL, N$
50 PRINT N$
60 NEXT I%
70 CLOSE #CAROL
```

We are still using the secretary analogy for the different file 'channels' that we introduced last time. Note the CLOSE# statement at the end. You must remember to close files when you've finished with them.

This program will work okay, and all the micros' names that you stored last time will be displayed on the screen. However, if you went back to last month's program that created the file in the first place, and added another computer to the list, to create a longer file, then this input program would have to be altered as well, to account for the longer list. This is because it uses a simple FOR-NEXT loop to read in the data. One solution would be to make the upper limit in the FOR-NEXT loop higher than the length of any list you would want to use. However this would cause an error when the program is run and your Electron tries to find data that isn't there.

A much more useful system would be one that would read in all of the data and stop when it gets to the end of the list. For this we need some method of detecting the end of a file. Fortunately there is just such a method provided in BBC Basic. We use the keyword EOF#.

EOF# stands for 'End Of File'. This is a function that returns a 'boolean' value (true or false, -1 or 0). If the end of the file has been reached then EOF# returns a value of true. Of course you must also specify the channel

number of the file you are investigating. Although you will only be able to open a single channel with a cassette system, Basic is written to cope with disc systems as well, so you have to refer to Carol again.

```
10 REM cassette file input demo 2
20 CAROL=OPENIN "MICROS"
30 REPEAT
40 INPUT #CAROL, N$
50 PRINT N$
60 UNTIL EOF #CAROL
70 CLOSE #CAROL
```

Now we are using a REPEAT-UNTIL loop to run through the list. This loop will continue to read in the names and display them on the screen until the end of your file is detected in line 60. Now your file containing the list of names can be any length you like, from one to hundreds. This program will cope with it unchanged.

Of course these programs that we have looked at so far are not really all that much use. All you can do with them is tediously store a load of names and then read them back and print them on the screen. Commercial uses of files - databases - have a lot more to them. We can make a small alteration to the last program to go towards this. We will change it to not just print out all the data willy nilly, but to look for a particular name:

```
10 REM cassette file input demo 3
20 MODE 6
30 INPUT TAB(10,10)"Name of micro ",
NAME$
40 CAROL=OPENIN "MICROS"
50 REPEAT
60 INPUT #CAROL, N$
70   IF  N$=NAME$  THEN  PRINT  N$;"
FOUND"
80 UNTIL EOF #CAROL
90 CLOSE #CAROL
```

This program is pretty much the same as the last, only now a string is input from the keyboard at the start and each time round the loop (each time a name is read in from the list file) a comparison is made between the name read in and the name previously entered. Only if the two match is anything displayed on the screen.

Of course this is still not a very useful system. All the program will tell you is if the name you have typed in is found in the list. What we want is for the program to tell you something you didn't know (or couldn't remember) about the computer you have named. For this we need to move onto files containing more than one field in each record.

The next program extends the idea of a database on home computers a little further. This program will set up a file containing facts and figures about any computers that you care to enter. Although the program, as it stands, is concerned only with home computers and only with certain aspects of them, it could easily be altered to deal with any subject.

```
10 REM simple 'database' creator
20 MODE 6
30 CH=OPENOUT "MICROS"
40 REPEAT
50 CLS
60 PRINT TAB(3,5)"Name of micro ";:I
NPUT name$
70 IF name$<>"*" THEN PROCinput
80 UNTIL name$="*"
90 CLOSE #CH
100 END
110 :
120 DEF PROCinput
130 PRINT TAB(3,7)"Memory capacity of
";name$;" (K) ";:INPUT memory%
140 PRINT TAB(3,9)"Display resolution
of ";name$;TAB(20,10)" X : ";:INPUT X%
150 PRINT TAB(20,11)" Y : ";:INPUT Y%
160 PRINT TAB(3,13)"Number of colours
";:INPUT colour%
170 CLS
180 PRINT TAB(2, 5);name$
190 PRINT memory%;"K"
200 PRINT X%;" X ";Y%;" graphics in "
;colour%;" colours."
210 PRINT TAB(2,10)"ARE THESE ALL COR
RECT?"
220 REPEAT
230 A%=INSTR("YyNn",GET$)
240 UNTIL A%
250 IF A%<3 THEN PRINT #CH,name$,memo
ry%,X%,Y%,colour%
260 ENDPROC
```

This program will create a file containing facts on home computers. It is a simple program but serves its purpose quite well. When you run the

program you are asked for each of a series of facts about a computer - name, memory size, graphics resolution, and number of colours. Then you are given a chance (lines 210 to 240) to change your mind about this data before it is stored. If you enter '*' as the computer name then the program will stop and close the file.

You will find that when you run this program, after entering the data for one micro, the cassette recorder doesn't spring into life and store the record as you might expect. This is due to the way in which all cassette data is handled by the Electron.

You will have noticed, when loading and saving Basic programs, that data is recorded onto cassette in 'blocks'. When data is saved to a cassette file, it is also split into blocks, each 256 bytes long. Each record of data corresponding to the information on one home computer does not make up a whole block. It is only a few bytes long.

What your Electron does then, when it comes across a PRINT# statement is not to send it direct to the cassette recorder but to store it in a section of memory called a buffer. Only when this buffer (256 bytes long) is full is the whole lot sent for recording onto tape. When you complete each record of data it is sent to this buffer. Only when you have entered enough, or when you finally close the file, is a whole block recorded onto the cassette.

Now we have created a simple database, we need a program to interrogate it, to extract the data that we want from it. The program to do this is really an extension of the program that searched for a particular name in our computer name list.

```
  10 REM simple 'database' interrogator
  20 DIM N%(5)
  30 MODE 6
  40 PRINT TAB(1,2)"SEARCH FOR:"
  50 PRINT TAB(14,4)"1.  Name"
  60 PRINT TAB(14,5)"2.  Memory"
  70 PRINT TAB(14,6)"3.  X Resolution"
  80 PRINT TAB(14,7)"4.  Y Resolution"
  90 PRINT TAB(14,8)"5.  No. Colours"
 100 PRINT TAB(1,12)"Input choice"
 110 REPEAT:PRINT TAB(13,12)SPC 5:INPU
T TAB(13,12),C%
 120 UNTIL C%<6 AND C%>0
 130 IF C%=1 THEN INPUT TAB(1,14)"name
",N$
 140 IF C%>1 THEN INPUT TAB(1,14)"Limi
ts: upper",U%:INPUT TAB(9,15)"lower",L%
 150 CLS
 160 CH=OPENIN "MICROS"
 170 PRINT''
 180 PRINT CHR$14
 190 REPEAT
 200 INPUT #CH,name$,N%(2),N%(3),N%(4)
,N%(5)
 210 IF C%=1 THEN IF name$=N$ THEN PRO
Cprint
 220 IF C%>1 THEN IF N%(C%)<U% AND N%(
C%)>L% THEN PROCprint
 230 UNTIL EOF #CH
 240 CLOSE #CH
 250 PRINT CHR$15
 260 END
 270 :
 280 DEF PROCprint
 290 PRINT name$
 300 PRINT N%(2);"K"
 310 PRINT N%(3);" x ";N%(4);" graphic
s in ";N%(5);" colours."
 320 PRINT ''
 330 ENDPROC
```

This program allows you to choose any one of a menu of search criteria. If one of the numerical search criteria (options 2 to 5) is chosen then upper and lower limits for these are also entered. If the first option is chosen then a computer name is input for the search. The file is read using a REPEAT-UNTIL EOF# loop much as before. If, in any record, the name matches up with the name input or the right piece of numerical data matches up then the whole record is printed out along with some suitable padding to make it readable (PROCprint). The numeric data is read into an array (N%) so that in line 220 the right data can be chosen according to the menu choice, C%.

Using this program you can search large files created with the last program. The file can be searched for computers with specifications that fall between particular limits, or the complete details on one computer can be displayed. As such, this program forms a simple but effective reference file. At the moment we have no way in which to add to, delete, or to alter records in our data file. We will look at ways of doing this next time.

CATERPILLAR

Dave Robinson

Caterpillar is a one player, fast moving action game in which you take the part of a caterpillar, trying to eat all of the bugs on the screen before they mature and become malicious.

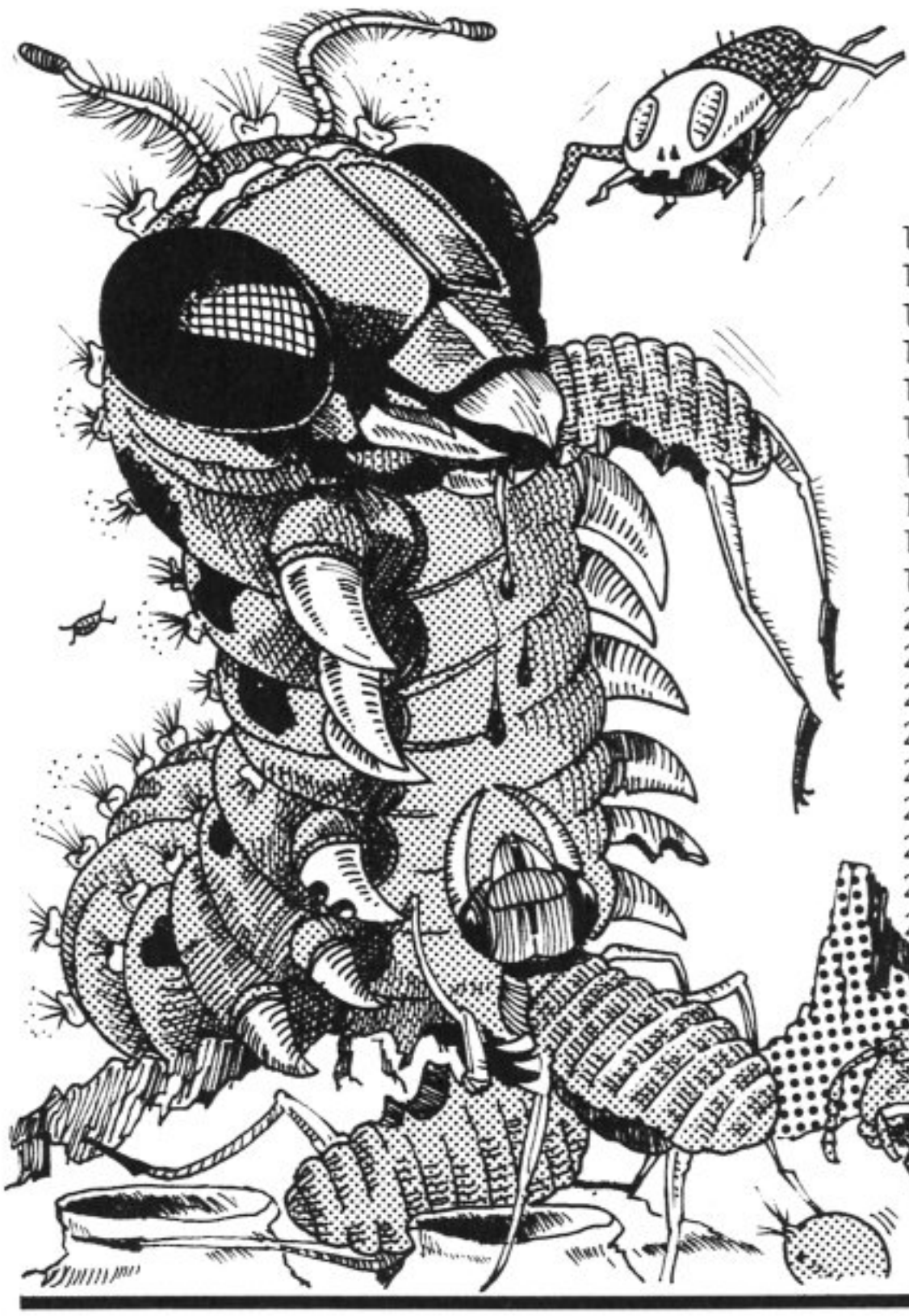
Your main aim is to complete three screens of action without being eaten. Each bug starts off as a small dot, and matures to a large bug. If this is not eaten quickly the bug transforms into a beastie and will start to chase and eat you.

The bugs evolve through three different stages; the small bug is worth 50 points, the medium one is worth 150 points and the large 'space ship lookalike' bug is worth 200 points. As you eat more bugs, the body length of the caterpillar becomes longer.

You lose points for every segment that is eaten by the beasties, but if you clear each sheet without encountering any beasties you are given a bonus score.

Full instructions are given in the program, and the keys that you use are 'Z' and 'X' for left and right, and '*' and '?' for up and down.

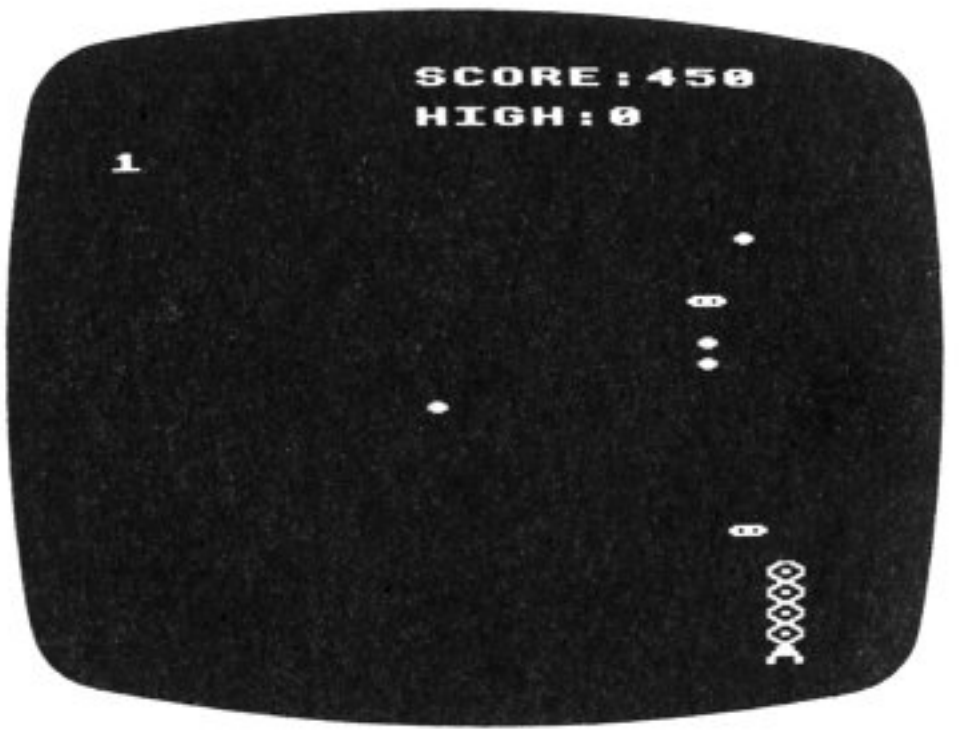
Once you become familiar with the basic pattern of play, it is possible to develop various strategies that both prolong the game and enhance your score. Happy eating!



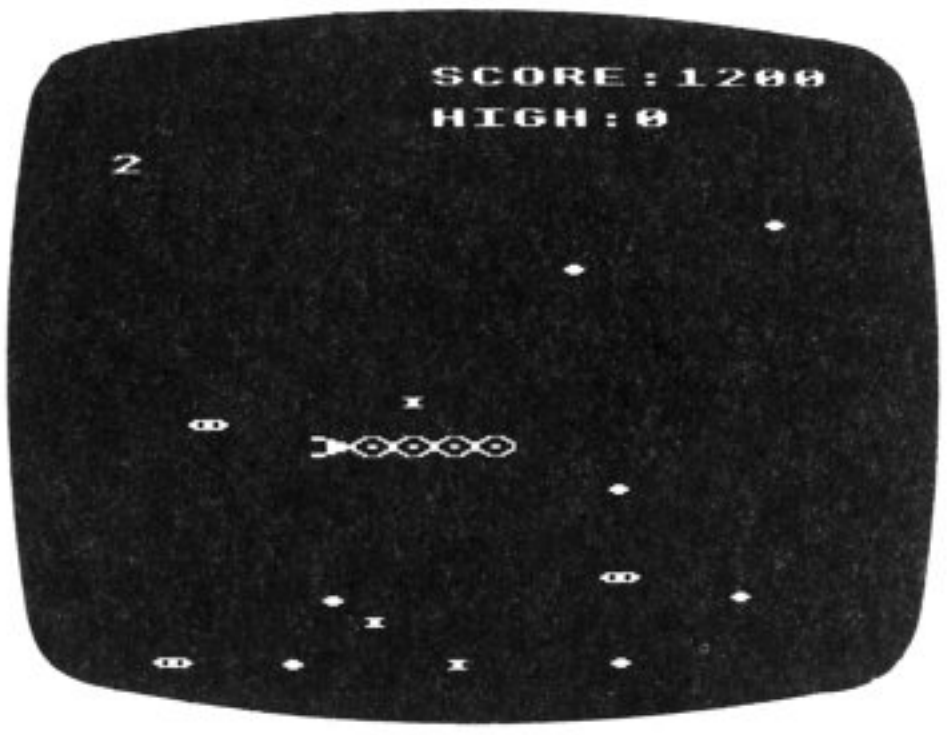
```
 10 REM PROGRAM CATERPILLAR
 20 REM Version E0.6
 30 REM Author  Dave Robinson
 40 REM ELBUG   JAN/FEB 1985
 50 REM Program Subject to copyright
 60 :
100 ON ERROR GOTO 400
110 DIM X%(21),Y%(21)
120 DIM head$(4),beast$(3)
130 DIM egg$(3),egg%(20)
140 DIM change%(20)
150 DIM BX%(20),BY%(20)
160 PROCinitial
170 MODE5
180 PROCinstruct
190 REPEAT
200 screen%=1
210 REPEAT
220 PROCsetup
230 PROCdrawscreen
240 REPEAT
250 PROCworm
260 IF move%>4 PROCchange:move%=0
270 IF aim%=target% won%=TRUE
280 PROCbeast
```


```
290 beast%=beast%MOD(BN%+1)+1
300 UNTIL won% OR lost%
 310 IF BN%=0 PROCbonus
 320 screen%=screen%+1
 330 UNTIL lost% OR screen%=4
 340 PROCpause(100)
 350 PROCending
 360 UNTIL NOT again%
 370 MODE6
 380 END
 390 :
 400 ON ERROR OFF
 410 MODE6
 420 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
 430 END
 440 :
1000 DEF PROCinitial
1010 VDU23,224,&3C,&66,&C3,&99,&99,&C3
,&66,&3C
1020 VDU23,225,&07,&1F,&78,&F8,&F8,&78
,&1F,&07
1030 VDU23,226,&E0,&F8,&1E,&1F,&1F,&1E
,&F8,&E0
1040 VDU23,227,&18,&3C,&3C,&7E,&7E,&C3
,&C3,&C3
1050 VDU23,228,&C3,&C3,&C3,&7E,&7E,&3C
,&3C,&3C
1060 VDU23,229,&0,&0,&18,&3C,&3C,&18,&
0,&0
1070 VDU23,230,&0,&0,&3C,&18,&18,&3C,&
0,&0
1080 VDU23,231,&0,&0,&7E,&DB,&DB,&7E,&
0,&0
1090 VDU23,232,&C3,&42,&7E,&DB,&DB,&7E
,&42,&C3
1100 VDU23,233,&66,&42,&7E,&DB,&DB,&7E
,&42,&66
1110 FOR I%=1 TO 4
1120 head$(I%)=CHR$17+CHR$2+CHR$(224+I
%)
1130 NEXT
1140 body$=CHR$17+CHR$2+CHR$224
1150 beast$(1)=CHR$17+CHR$1+CHR$232
1160 beast$(2)=CHR$17+CHR$1+CHR$233
1170 beast$(3)=CHR$17+CHR$2+CHR$233
1180 FORI%=1 TO 3
1190 egg$(I%)=CHR$17+CHR$3+CHR$(228+I%
)
1200 NEXT
1210 hiscore%=0:score%=0
1220 ENDPROC
1230 :
1240 DEF PROCsetup
1250 BN%=0
1260 lost%=FALSE:won%=FALSE
1270 FOR I%=0 TO 5
1280 X%(I%)=I%+1:Y%(I%)=10
1290 NEXT
1300 OX%=5:OY%=10:head%=4:tail%=0:aim%
=-1:B%=1
1310 FOR I%=0 TO 20
1320 BX%(I%)=0:BY%(I%)=0
1330 NEXT
1340 target%=4*screen%+4:WXdir%=+1:WYd
ir%=0:H%=1:move%=0:beast%=0
1350 ENDPROC
1360 :
1370 DEF PROCdrawscreen
1380 CLS:VDU23,1,0;0;0;0;
1390 VDU19,2,6,0,0,0:VDU19,1,3,0,0,0
1400 PRINTTAB(9,1)"SCORE:"
1410 PRINTTAB(9,3)"HIGH:"
1420 PRINTTAB(1,5);screen%
1430 GCOL0,1:MOVE32,48
1440 DRAW1248,48:DRAW1248,816
1450 DRAW32,816:DRAW32,48
1460 FOR I%=1 TO 4
1470 PRINTTAB(I%,10)body$
1480 NEXT
1490 PRINTTAB(5,10)head$(1)
1500 FOR I%=0 TO target%
1510 REPEAT
1520 J%=RND(16)+1:K%=RND(22)+7
1530 UNTIL FNpoint(J%,K%)=0
1540 PRINTTAB(J%,K%)egg$(1)
1550 egg%(I%)=J%*&100+K%
1560 change%(I%)=1
1570 NEXT
1580 FOR I%=target%+1 TO 20
1590 egg%(I%)=0:change%(I%)=4
1600 NEXT
1610 COLOUR2:PRINTTAB(15,1);score%
1620 PRINTTAB(14,3);hiscore%
1630 ENDPROC
1640 :
1650 DEF PROCworm
1660 move%=move%+1:X%=OX%:Y%=OY%
1670 IF INKEY(-67) WXdir%=+1:WYdir%=0:
H%=1
1680 IF INKEY(-98) WXdir%=-1:WYdir%=0:
H%=2
1690 IF INKEY(-105)WYdir%=+1:WXdir%=0:
H%=3
```

```
 1700 IF INKEY(-73) WYdir%=-1:WXdir%=0:
H%=4
 1710 X%=X%+WXdir%:Y%=Y%+WYdir%
 1720 C%=FNpoint(X%,Y%)
 1730 IF C%=3 PROCswallow:ENDPROC
 1740 IF C%<>0 ENDPROC
 1750 PRINTTAB(X%,Y%)head$(H%)
 1760 PRINTTAB(X%(head%),Y%(head%))body
$
 1770 PRINTTAB(X%(tail%),Y%(tail%))CHR$
32
 1780 OX%=X%:OY%=Y%
 1790 head%=(head%+1)MOD21
 1800 tail%=(tail%+1)MOD21
 1810 X%(head%)=X%:Y%(head%)=Y%
 1820 SOUND1,-15,(X%+10)*Y%,1
 1830 ENDPROC
 1840 :
 1850 DEF PROCbeast
 1860 IF BN%=0 PROCpause(3)
 1870 IF BX%(beast%)=0 ENDPROC
 1880 OBX%=BX%(beast%):OBY%=BY%(beast%)
 1890 Xdir%=SGN(X%(tail%)-OBX%)
 1900 Ydir%=SGN(Y%(tail%)-OBY%)
 1910 IF OBX%+Xdir%=X%(tail%) AND OBY%+
Ydir%=Y%(tail%)PROCeat:ENDPROC
 1920 IF FNpoint(OBX%+Xdir%,OBY%+Ydir%)
<>0 ENDPROC
 1930 BX%(beast%)=OBX%+Xdir%:BY%(beast%
)=OBY%+Ydir%
 1940 PRINTTAB(BX%(beast%),BY%(beast%))
beast$(B%)
 1950 PRINTTAB(OBX%,OBY%)CHR$32
 1960 B%=(B%MOD2)+1
 1970 ENDPROC
 1980 :
 1990 DEF PROCchange
 2000 Z%=RND(21)-1
 2010 IF change%(Z%)=4 OR egg%(Z%)=0 EN
DPROC
 2020 IF change%(Z%)=3 PROCnewbeast:END
PROC
 2030 PRINTTAB(egg%(Z%)DIV&100,egg%(Z%)
MOD&100)egg$(change%(Z%)+1)
 2040 IF change%(Z%)<3 change%(Z%)=chan
ge%(Z%)+1
 2050 ENDPROC
 2060 :
 2070 DEF PROCnewbeast
 2080 PRINTTAB(egg%(Z%)DIV&100,egg%(Z%)
MOD&100)beast$(1)
 2090 change%(Z%)=4
 2100 target%=target%-1:BN%=BN%+1
 2110 BX%(BN%)=egg%(Z%)DIV&100:BY%(BN%)
=egg%(Z%)MOD&100
 2120 ENDPROC
 2130 :
 2140 DEF PROCbonus
 2150 COLOUR1:bonus%=screen%*500
 2160 PRINTTAB(1,2)"*BONUS*"
```




```
 2170 PRINTTAB(3,4);bonus%
 2180 PROCpause(50)
 2190 score%=score%+bonus%
 2200 COLOUR2:PRINTTAB(15,1);score%
 2210 PROCpause(100)
 2220 ENDPROC
 2230 :
 2240 DEF PROCending
 2250 *FX15,1
 2260 again%=TRUE:CLS:COLOUR1
 2270 IF lost% PRINTTAB(6,6)"OH DEAR!",
TAB(6,8)"YOU LOST"
 2280 IF won% PRINTTAB(5,8)"WELL DONE!"
 2290 PRINTTAB(3,11)STRING$(14,"-")
 2300 COLOUR2
 2310 PRINTTAB(5,15)"SCORE:";score%
 2320 IF score%>hiscore% THEN hiscore%=
score%
 2330 PRINTTAB(5,18)"HIGH:";hiscore%
 2340 COLOUR3
 2350 PRINTTAB(4,24)"AGAIN (Y/N)?"
 2360 REPEAT:UNTIL INKEY-69 OR INKEY-86
 2370 IF GET$="N"again%=FALSE
 2380 score%=0
 2390 ENDPROC
 2400 :
 2410 DEF PROCswallow
 2420 IF head%<tail% top%=head%+21 ELSE
 top%=head%
 2430 PRINTTAB(X%,Y%)head$(H%)
 2440 S%=-1
 2450 REPEAT
 2460 S%=S%+1
 2470 UNTIL egg%(S%)DIV&100=X% AND egg%
(S%)MOD&100=Y%
 2480 egg%(S%)=0
 2490 PRINTTAB(X%(head%),Y%(head%))body
$
 2500 FOR I%=top% TO tail% STEP-1
 2510 PRINTTAB(X%(I%MOD21),Y%(I%MOD21))
egg$(1)
 2520 SOUND1,-15,1*I%*3,1
 2530 PROCpause(2)
```




```
 2540 PRINTTAB(X%(I%MOD21),Y%(I%MOD21))
body$
 2550 NEXT
 2560 OX%=X%:OY%=Y%
 2570 head%=(head%+1)MOD21
 2580 X%(head%)=OX%:Y%(head%)=OY%
 2590 aim%=aim%+1
 2600 score%=score%+change%(S%)*50
 2610 COLOUR2:PRINTTAB(15,1);score%
 2620 PRINTTAB(15,1);score%
 2630 IF aim%=target% won%=TRUE
 2640 ENDPROC
 2650 :
 2660 DEF PROCeat
 2670 PRINTTAB(OBX%+Xdir%,OBY%+Ydir%)be
ast$(3)
 2680 tail%=(tail%+1)MOD21
 2690 aim%=aim%-1
 2700 IF aim%=-4 THEN lost%=TRUE
 2710 PRINTTAB(OBX%,OBY%)CHR$32
 2720 SOUND1,-15,10,2
 2730 PROCpause(10)
 2740 SOUND1,-15,60,1
 2750 PRINTTAB(OBX%,OBY%)egg$(1)
 2760 S%=-1
 2770 REPEAT:S%=S%+1:UNTIL egg%(S%)=0
 2780 egg%(S%)=OBX%*&100+OBY%
 2790 change%(S%)=1
 2800 PRINTTAB(OBX%+Xdir%,OBY%+Ydir%)be
ast$(B%)
 2810 BX%(beast%)=BX%(beast%)+Xdir%:BY%
(beast%)=BY%(beast%)+Ydir%
 2820 score%=score%-50
 2830 COLOUR2:PRINTTAB(15,1);score%
 2840 ENDPROC
 2850 :
 2860 DEFFNpoint(S%,T%)
 2870 =POINT(S%*64+32,1008-T%*32)
 2880 :
 2890 DEF PROCpause(time%)
 2900 TIME=0:REPEAT:UNTIL TIME=time%
 2910 ENDPROC
 2920 :
 2930 DEF PROCinstruct
 2940 VDU19,1,6,0,0,0
 2950 VDU23,1,0;0;0;0;
 2960 COLOUR1
 2970 PRINTTAB(4,2)"CATERPILLAR"
 2980 PRINTTAB(4,3)STRING$(11,"-")
 2990 COLOUR3
 3000 PRINT'"EAT THE EGGS BEFORE"
 3010 PRINT'"THEY EVOLVE INTO .."
 3020 COLOUR2
 3030 PRINT'TAB(1)"HUNGRY BEASTIES! ";C
HR$232
 3040 COLOUR3
 3050 PRINTTAB(3,12);CHR$229;SPC3;"50 p
oints"
 3060 PRINTTAB(3,15);CHR$230;SPC2"150 p
oints"
 3070 PRINTTAB(3,18);CHR$231;SPC2"200 p
oints"
 3080 COLOUR1:PRINT''"CONTROLS:"
 3090 COLOUR3
 3100 PRINT''TAB(2)"Z LEFT.   * UP"
 3110 PRINT'TAB(2)"X RIGHT.  ? DOWN"
 3120 COLOUR2
 3130 PRINTTAB(3,30)"SPACE TO START"
 3140 REPEAT
 3150 PRINTTAB(17,9)CHR$232,TAB(16,9)CH
R$32
 3160 IF INKEY(-99) ENDPROC
 3170 PROCpause(30)
 3180 PRINTTAB(16,9)CHR$233,TAB(17,9)CH
R$32
 3190 IF INKEY(-99) ENDPROC
 3200 PROCpause(30)
 3210 UNTIL FALSE
 3220 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINT

QUICK WAIT FOR KEY - D. Morgan

When writing 'user friendly' programs, it is often necessary to wait for a key to be pressed in order to provide time for the user to read a piece of text. The most memory efficient way in which Basic can perform this is by means of the following:

```
IF GET
```

This uses much less memory than the more obvious A=GET. Brief, ain't it!

Subscriptions and Software Help Line
St Albans (0727) 60263
Manned Mon-Fri 9.30am-4.30pm.

BACK ISSUES AND SUBSCRIPTIONS

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ELBUG members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks
HP10 8HQ

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for one year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

SOFTWARE (Members only)

This is available from the software address.

MAGAZINE CONTRIBUTIONS AND TECHNICAL QUERIES

Please send all contributions and technical queries to the editorial address opposite. All contributions published in the magazine will be paid for at the rate of £25 per page.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG
PO Box 50
St Albans
Herts

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Assistant Editor: Geoff Bains. Production Editor: Phyllida Vanstone.
Technical Assistant Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to, Sheridan Williams, and Adrian Calcraft for assistance with this issue.

New Elbug Binders

We have produced an attractive hard-backed binder for the ELBUG magazine. These binders are green in colour with "ELBUG" in gold lettering on the spine and allow for the whole of one volume of the magazine to be stored as a single reference book.

Each binder will accommodate 10 ELBUG magazines, and is supplied with 12 wires to enable the index and the latest copy of the supplement to be included within the binder if required. Individual issues may be easily added and removed, allowing for the latest volume to be filed as it arrives.

Elbug Binders

NEW YEAR SPECIAL OFFER

£1.90

If ordered before March 1st

The price of the new ELBUG binder is £3.90 including VAT, please add 50p post and packing for delivery within the U.K. Overseas members please send the same amount, this will cover the extra postage but not VAT.
Plese send to:
BEEBUGSOFT, PO BOX 109, High Wycombe, Bucks, HP10 8HQ.

THE BEST OF ELBUG ON CASSETTE

Many of the best programs published in ELBUG have been collected together and published by Penguin Books under the name "Games and other programs for the Acorn Electron" at £3.95. This book is part of the Penguin Acorn Computer Library and at present there is just one other title available though others are planned.

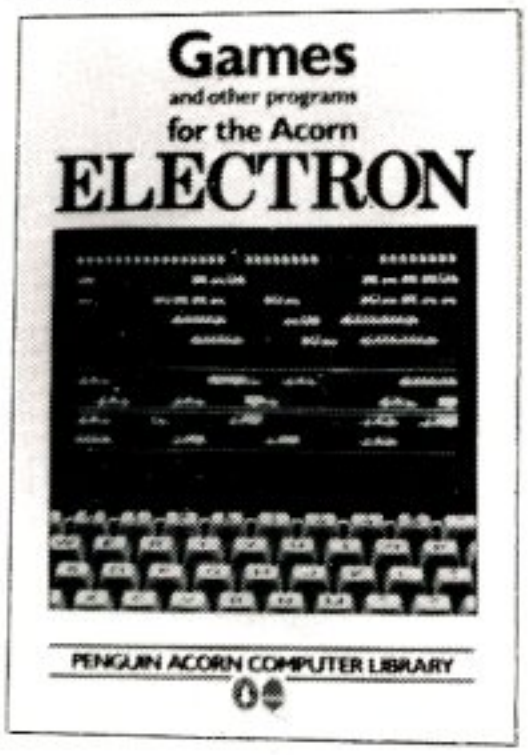
There are 20 programs in all in four different categories:

Action Games
- Munch-Man, Mars Lander, Invasion
- Robot Attack, Hedgehog

Thought games
- Higher/Lower, Five-Dice, Life
- Anagrams, Return of the Diamond

Visual Displays
- Union Jack, Square Dance, Ellipto
- Screenplay, 3-D Rotation

Utilities
- Sound Wizard, Bad Program Lister
- 3-D Lettering, Bad Program Rescue
- Double Height Text



All 20 programs are now available on cassette from our software address (in High Wycombe) price £7 to members and £9 to non-members, plus 50p post & packing in either case.

ELBUG MAGAZINE CASSETTE

To save wear and tear on fingers and brain, we offer, each month, a cassette of the programs featured in the latest edition of ELBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles.

Magazine cassettes have been produced for each issue of ELBUG from Volume 1 Number 1 onwards and are all available from stock, priced £3.00 each inclusive of VAT. See below for ordering information.

This months cassette includes:

Volume 2 Number 3

Caterpillar (a fast moving action game), fascinating 3D surfaces program, a greatly extended Mini Text Editor, a program to illustrate Making Tunes on the Electron, Patience (a very good implementation of this popular card game), a Trace Facility for help in debugging programs, and all the example programs from this month's article on File Handling.

MAGAZINE CASSETTE SUBSCRIPTION

We are also able to offer ELBUG members subscription to the magazine cassette, this gives the added advantage of receiving the cassette at around the same time as the magazine each month. Subscriptions may either be for a period of 1 year or 6 months. (NOTE Magazine cassettes are produced 10 times each year).

If required, subscriptions may be backdated as far as Volume 1 Number 1, so when applying please write to the address below quoting your membership number and the issue from which you would like your subscription to start.

MAGAZINE CASSETTE ORDERING INFORMATION

Individual ELBUG Magazine Cassettes £3.00.
P & P: Please add 50p for the first and 30p for each subsequent cassette.
Overseas orders: Please send the same amount, this will include the extra post but not VAT.

Magazine Cassette Subscription

1 YEAR (10 issues)	£33.00	Inclusive.....	O'SEAS	£39.00 No VAT payable
6 MONTHS (5 issues)	£17.00	Inclusive.....	O'SEAS	£20.00 No VAT payable

Please be sure to specify that you require subscription to the ELBUG magazine cassette (as opposed to the BEEBUG cassette), and enclose your membership number with a cheque made payable to BEEBUGSOFT.

Please send to..

ELBUG Magazine Cassette, BEEBUGSOFT, PO Box 109, High Wycombe, HP10 8HQ

Printed in England by Staples Printers St Albans Limited at The Priory Press.